НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено

Завідувач кафедри

_____ О.В. Коваль

(підпис) (ініціали, прізвище)

“___”_____2019р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 “Програмна інженерія”

на тему “Система моделювання мінімальних поверхонь на основі ізотропних кривих з квазіконформною заміною параметра”

Виконав: студент IV курсу, групи ТІ-51

Кондратьєв Арсен Сергійович _____

(прізвище, ім’я, по батькові) (підпис)

Керівник ст.викладач, Гурін Артем Леонідович _____

(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент_____ _____

(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

(підпис)

Київ — 2019

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший , бакалаврський

Напрям підготовки 6.050103 “ Програмна інженерія”

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ О.В. Коваль

(підпис)

”___”_____2019р.

ЗАВДАННЯ

на дипломну роботу студенту

Кондртьєву Арсену Сергійовичу

(прізвище, ім’я, по батькові)

1. Тема роботи “Система моделювання мінімальних поверхонь на основі ізотропних кривих з квазіконформною заміною параметра”

керівник роботи Гурін Артем Леонідович, ст. викладач

(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від 22.05.2019р. № 1325-с

2. Строк подання студентом роботи 10 червня 2019р.

3. Вихідні дані до роботи розроблений модуль написаний мовою Python

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити) виконати аналіз існуючих систем моделювання мінімальних поверхонь, створити власний програмний додаток для реалізації задачі моделювання мінімальних поверхонь на основі ізотропних кривих з квазіконформною заміною параметра.

5. Перелік ілюстративного матеріалу титульний аркуш, поставлені задачі, функції розроблюваної системи, необхідні формули для реалізації задачі

моделювання мінімальних поверхонь, архітектура системи, блок-схеми та uml-діаграми розробленої системи, детальний опис модулів додатку, ілюстративний матеріал роботи користувача з програмною системою, висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання "_10_" ___жовтня_____2018__р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі	12.02.19	
2	Розробка архітектури та загальної структури системи	23.04.19	
3.	Реалізація головного модуля моделювання мінімальних поверхонь	30.04.19	
4.	Програмна реалізація системи	02.05.19	
5.	Оформлення пояснювальної записки	10.05.19	
6.	Захист програмного продукту	18.05.19	
7.	Передзахист	01.06.19	
8.	Захист	19.06.19	

Студент _____ Кондратьєв А. С.____

(підпис) (прізвище та ініціали,)

Керівник роботи _____ ____ст.викладач, Гурін А. Л.___

(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Дипломну роботу виконано на 64 аркушах, вона містить 3 додатки та перелік посилань на використані джерела з 19 найменувань. У роботі наведено 23 рисунки та 3 таблиці.

В процесі виконання роботи було порівняно та проаналізовано існуючі засоби створення мінімальних поверхонь, зроблено огляд існуючого програмного продукту для моделювання поверхнонь, K3DSurf.

Результатом роботи є створений програмний застосунок, що надає вичерпний та інтуїтивно зрозумілий користувацький інтерфейс для моделювання мінімальних поверхонь, де у якості напрямної кривої використовується кубічна ізотропна крива Безьє 3-го порядку.

Ключові слова: мінімальна поверхня, ізотропна крива, кривизна поверхні, квазіконформна заміна параметру.

ANNOTATION

Graduate work was completed on 64 sheets, it contains 3 applications and a list of references to used sources of 19 titles. The paper presents 23 drawings and 3 tables.

During the work, existing means of creation of minimal surfaces were compared and analyzed, overview of the existing software K3DSurf for surface modeling is madeThe result of the work is a software application that provides an exhaustive and intuitive user interface for modeling the minimal surfaces, where as a guide curve, the cubic isotropic Bezier curve of the 3rd order is used.

Keywords: minimal surface, isotropic curve, surface curvature, quasiconformal parameter change.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

MVC

Мінімальна поверхня

Кривина

Ізотропна крива

Квазіконформна заміна параметру

ВСТУП

З кожним роком набувають все більшого поширення різноманітні програмні продукти та бібліотеки різних мов програмування для графічного моделювання. На їх основі вже побудовано багато різного програмного забезпечення, яке вирішує широкий спектр прикладних задач. Серед них можна виділити і задачу моделювання мінімальних поверхонь.

Мінімальна поверхня - гладка поверхня з нульовою середньою кривизною. Назва пояснюється тим, що у гладкої поверхні із заданим контуром мінімізуюча площа є мінімальною.

У диференціальній геометрії, кривина — збірна назва ряду кількісних характеристик (чисельних, векторних, тензорних), що описують відхилення того або іншого геометричного «об'єкта» (кривої, поверхні, рімановового простору тощо) від відповідних «пласких» об'єктів (пряма, площина, евклідів простір тощо).

Наразі вже існує декілька програмних додатків, що дозволяють здійснювати моделювання мінімальних поверхонь, але загалом область застосування та їх загальне призначення є набагато ширшим.

Основним недоліком існуючих систем є їх громіздкість, спричинена тим, що вони набагато ширшого призначення. Для початківців ці програми можуть здатися переповненими функціоналом та дуже специфічними в плані налаштування.

Можна відзначити такі застосунки, як K3DSurf, Maple, але вони є доволі громіздкими, їх функціонал та можливості виходять далеко за межі одного лише моделювання мінімальних поверхонь, також більшість таких застосунків має лише англійську локалізацію, що вимагає від користувача додаткових знань. Тому було вирішено розробити зручний, не перевантажений зайвим функціоналом та можливостями, не вибагливий до машинних ресурсів програмний застосунок з наявною українською локалізацією для моделювання саме мінімальних поверхонь. Також можна відзначити, що ефективність та швидкість отримання результатів, при використанні такого застосунку буду набагато вищою.

Програмний додаток розроблявся з використанням мови програмування Python, платформи PyQt5 та бібліотеки Plotly.py, призначеної для інтерактивного графічного моделювання.

У завдання дипломного проекту входять:

1. Ознайомлення з існуючими системами та програмними застосунками для здійсненння моделювання мінімальних поверхнонь
2. Створення програмного модулю для моделювання мінімальних поверхнонь, де, напрямна задається аналітичною функцією
3. Розробка графічного інтерфейсу для взаємодії користувача зі створеною програмною системою.
4. Перевірка створеної поверхні на мінімальність за допомогою переврки середньої кривизни поверхні, повпередньо розрахованої з використанням коефіцієнтів першої та другої квадратичних форм
5. Збереження отриманих змодельованих поверхонь у зручному форматі
6. Здійснено локалізацію програмного додатку українською мовою

Дипломна робота складається з: вступу, п’яти розділів, списку використаних джерел та трьох додатків.

1. ПОСТАНОВКА ЗАДАЧІ МОДЕЛЮВАННЯ МІНІМАЛЬНИХ ПОВЕРХОНЬ НА ОСНОВІ ІЗІТРОПНИХ КРИВИХ З КВАЗІКОНФОРМНОЮ ЗАМІНОЮ ПАРАМЕТРА

Метою роботи є дослідження методів моделювання мінімальних поверхонь та розробка програмного застосунка, що має моделювати та відображувати мінімальні поверхні, створені на основі ізотропних кривих з квазіконформною заміною параметра.

Призначенням даного програмного застосунка є надання зручного та вичерпного користувацького інтерфейсу для реалізації задачі моделювання мінімальних поверхонь. Програмний засіб призначений для використання на комп’ютерах, що використовують операційну систему Windows XP/7/10.

Вхідні дані: коефіцієнти аналітичної функції, за домогою яких буде здійснено розрахунок координат характеристичного чотирикутнига ізотропної кривої Безьє третього порядку.

Вихідні дані: Змодельована мінімальна поверхня (поверхні) у форматі .html.

Система має відповідати даним вимогам:

1. Реалізовувати основний функціонал для здійснення моделювання мінімальних поверхонь
2. Робити точні розрахунки та графічне моделювання відповідно до заданих параметрів.
3. Здійснювати перевірки заданих значень щодо правильності введення
4. Здійснювати перевірку заданої поверхні на мінімальність, розраховуючи та перевіряючи значення середньої кривизни поверхні, розраховане за допомогою коефіцієнтів першої та другої квадратичних форм
5. Мати вичерпний та інтуїтивно зрозумілий користувацький інтерфейс
6. Зберігати змодельовані мінімальні поверхні в зручному форматі

2. АНАЛІЗ ПРОБЛЕМИ МОДЕЛЮВАННЯ МІНІМАЛЬНИХ ПОВЕРХОНЬ

2.1. Опис предметної області

Теорія мінімальної поверхні походить з Лагранжа, який у 1762 р. Розглянув варіаційну задачу знаходження поверхні z = z (x, y) найменшої площі, протяжної через заданий замкнутий контур. Він отримав рівняння Ейлера – Лагранжа, але він не зміг знайти жодного рішення за межами літака. У 1776 році Жан Батіст Марі Межньє виявив, що гелікоиди і катеноїди задовольняють рівняння і що диференціальне вираження відповідає подвійній середній кривизні поверхні, роблячи висновок, що поверхні з нульовою середньою кривизною мінімізують область.Гаспар Монж і Лежандр у 1795 р. Отримали формули формулювання для поверхонь розчину. Хоча вони були успішно використані Генріхом Щерком в 1830 році для виведення його поверхонь, вони, як правило, вважалися практично непридатними для використання. У 1842/43 каталонці довели, що гелікоида є єдиною мінімальною поверхнею.

Прогрес був досить повільним до середини століття, коли проблему Бйорлінга вирішували за допомогою складних методів. Почався «перший золотий вік» мінімальних поверхонь. Шварц знайшов рішення проблеми Плато для регулярного чотирикутника в 1865 р. І для загального чотирикутника в 1867 р. (Що дозволяє побудувати його періодичні поверхневі сім'ї), використовуючи складні методи. Вейерштрасс і Еннепер розробили більш корисні формули представлення, міцно зв'язуючи мінімальні поверхні зі складним аналізом і гармонійними функціями. Інші важливі внески прийшли від Beltrami, Bonnet, Darboux, Lie, Riemann, Serret і Weingarten.

Між 1925 і 1950 рр. Відновилася мінімальна поверхнева теорія, в даний час головним чином спрямована на непараметричні мінімальні поверхні. Повне вирішення проблеми плато Джессом Дугласом і Тібор Радо стало важливою віхою. Проблема Бернштейна і робота Роберта Оссермана щодо повних мінімальних поверхонь кінцевої загальної кривизни також були важливими.

Інше відродження почалося в 1980-х роках. Однією з причин було відкриття в 1982 році Celso Costa поверхні, яка спростувала гіпотезу про те, що площина, катеноїд і гелікоїд є єдиними повними мінімальними поверхнями в R3 кінцевого топологічного типу. Це не тільки стимулювало нову роботу щодо використання старих параметричних методів, але й продемонструвало важливість комп'ютерної графіки для візуалізації досліджуваних поверхонь і чисельних методів для вирішення "періодичної проблеми" (при використанні методу сполученої поверхні для визначення поверхневих латок, які можуть бути зібрані в більшу симетричну поверхню, деякі параметри повинні бути чисельно узгоджені для отримання вбудованої поверхні). Іншою причиною була перевірка Х. Карчером, що трипо періодичні мінімальні поверхні, що спочатку описувалися емпірично Аланом Шеном в 1970 році, насправді існують. Це призвело до багатого звіринця поверхневих сімей і методів отримання нових поверхонь від старих, наприклад, шляхом додавання ручок або спотворення їх.

В даний час теорія мінімальних поверхонь має різноманітність до мінімальних підмноговидностей в інших геометріях навколишнього середовища, стаючи актуальними для математичної фізики (наприклад, позитивна масова гіпотеза, гіпотеза Пенроуза) і тривимірної геометрії (наприклад, гіпотеза Сміта, гіпотеза Пуанкаре, геометрія Терстона Гіпотеза).

До класичних прикладів мінімальних поверхонь належать:

- площини, що є тривіальним випадком

- катеноїди: мінімальні поверхні, зроблені обертанням контактної мережі один раз навколо своєї прямій
- гелікоїди: поверхні, винесена лінією, що обертається з рівномірною швидкістю навколо осі, перпендикулярної до лінії, і одночасно рухається вздовж осі з однаковою швидкістю

Поверхні Золотого століття 19-го століття включають:

- Мінімальні поверхні Шварца: потрійно періодичні поверхні, що заповнюють R3
- Мінімальна поверхня Рімана: посмертно описана періодична поверхня
- поверхню Enneper
- поверхню Хеннеберга: перша не орієнтована мінімальна поверхня
- Мінімальну поверхню Бура

До сучасних поверхонь належать:

- Gyroid: одна з поверхонь Schoen 1970 року, трипільна періодична поверхня, що представляє особливий інтерес для структури рідких кристалів
- Сімейна башта: узагальнення другої поверхні Шерка
- Мінімальна поверхня Кости: відома гіпотеза позбавлена. Описаний в 1982 році Селсо Коста, а потім візуалізується Джимом Хоффманом. Джим Хоффман, Девід Хоффман і Вільям Мікс розширили визначення для створення сімейства поверхонь з різними ротаційними симетріями.
- поверхня сім'ї Чен – Гакстеттер, що додає ручки до поверхні Еннепера.

Мінімальні поверхні можуть бути визначені в інших різноманіттях, ніж R3, такі як гіперболічний простір, вищі розмірні простори або ріманові різноманіття.

Визначення мінімальних поверхонь може бути узагальнене / розширене для покриття поверхонь з постійною середньою кривизною: поверхонь з постійною середньою кривизною, які не повинні дорівнювати нулю.

У дискретній диференціальній геометрії вивчаються дискретні мінімальні поверхні: симпліціальні комплекси трикутників, які мінімізують їх площу під малими збуреннями їх вершинних положень. Такі дискретизації часто використовуються для апроксимації мінімальних поверхонь чисельно, навіть якщо не відомі вирази закритої форми. Броунівський рух на мінімальній поверхні призводить до імовірнісних доказів декількох теорем на мінімальних поверхнях.

Мінімальні поверхні стали зоною інтенсивного наукового дослідження, особливо в області молекулярної інженерії та матеріалознавства, завдяки їх очікуваним застосуванням у самостійному складенні складних матеріалів. запропоновано перебувати під еволюційним тиском, щоб відповідати нетривіальної мінімальної поверхні. Мінімальні поверхні грають роль у загальній теорії відносності. Відомий горизонт (незначно зовнішня захоплена поверхня) є мінімальною гіперповерхнею, що зв'язує теорію чорних дірок з мінімальними поверхнями і проблему Плато.

2.2. Моделювання поверхонь на основі ізотропних сіток

Поверхня в комплексному просторі визначатиметься кінематичним методом завдяки руху однієї ізотропної кривої (твірної) за іншими заданими ізотропними кривими (напрямними).

Виділивши окремо дійсну та уявну частини, матимемо дві поверхні для дослідження.

$$x_{\mathrm{Re}}(u,v)=\mathrm{Re}(x(u,v)),\ \ y_{\mathrm{Re}}(u,v)=\mathrm{Re}(y(u,v)),\ \ z_{\mathrm{Re}}(u,v)=\mathrm{Re}(z(u,v));$$
$$x_{\mathrm{Im}}(u,v)=\mathrm{Im}(x(u,v)),\ \ y_{\mathrm{Im}}(u,v)=\mathrm{Im}(y(u,v)),\ \ z_{\mathrm{Im}}(u,v)=\mathrm{Im}(z(u,v)). \qquad (2.1)$$

Застосовуючи метод Вейерштрасса, будуємо поверхню на основі направляючої ізотропної кривої та конформної

(t = u+v) або квазіконформної (t = ku + iv або t = u + ikv) заміни параметру:

$$x_{\mathrm{Re}}(u,v) = \mathrm{Re}(x(t)),\ y_{\mathrm{Re}}(u,v) = \mathrm{Re}(y(t)),\ z_{\mathrm{Re}}(u,v) = \mathrm{Re}(z(t));$$
$$x_{\mathrm{Im}}(u,v) = \mathrm{Im}(x(t)),\ y_{\mathrm{Im}}(u,v) = \mathrm{Im}(y(t)),\ z_{\mathrm{Im}}(u,v) = \mathrm{Im}(z(t)). \qquad (2.2)$$

де $x = x(t),\ y = y(t),\ z = z(t)$ - просторова параметрична ізотропна крива.

При конформній та квазікомфорній заміні параметру виділення дійсної та уявної частини дозволяє одержувати мінімальні та приєднані мінімальні поверхні у дійсному просторі. Якщо в рівнянні поверхні (2.2) вирази $x = x(t),\ y = y(t),\ z = z(t)$ визначають плоску ізотропну криву, тоді будемо мати ізотропну сітку на площині.

Для дослідження поверхонь будемо розраховувати коефіцієнти першої квадратичної форми:

$$F = x_u(u,v)x_v(u,v) + y_u(u,v)y_v(u,v) + z_u(u,v)z_v(u,v), \qquad (2.3)$$
$$E = x_u(u,v)^2 + y_u(u,v)^2 + z_u(u,v)^2,$$
$$G = x_v(u,v)^2 + y_v(u,v)^2 + z_v(u,v)^2,$$

та другої квадратичної форми:

$$L = \frac{1}{\sqrt{EG - F^2}}\begin{vmatrix} x_{uu} & y_{uu} & z_{uu} \\ x_u & y_u & z_u \\ x_v & y_v & z_v \end{vmatrix},\ M = \frac{1}{\sqrt{EG - F^2}}\begin{vmatrix} x_{uv} & y_{uv} & z_{uv} \\ x_u & y_u & z_u \\ x_v & y_v & z_v \end{vmatrix}, \qquad (2.4)$$
$$N = \frac{1}{\sqrt{EG - F^2}}\begin{vmatrix} x_{vv} & y_{vv} & z_{vv} \\ x_u & y_u & z_u \\ x_v & y_v & z_v \end{vmatrix}.$$

А також розраховувати середню кривину для поверхні:

$$H = \frac{1}{2}\frac{LG - 2MF + NE}{EG - F^2}. \qquad (2.5)$$

2.3. Моделювання дійсних мінімальних поверхонь на основі кривої Без'є

Візьмемо у якості напрямної кривої - криву Без'є та підставимо у рівняння (2.2). Будемо мати:

$$\mathbf{r}_{\mathrm{Re}}(u,v) = \mathrm{Re}(\sum_{j=0}^{n} \mathbf{r}_j J_{n,j}(u+iv)), \tag{2.6}$$

$$J_{n,j}(u+iv) = \frac{n!}{j!(n-j)!} t^j (1-u-iv)^{n-j},$$

де $\mathbf{r}_j = \lfloor x_{j\,\mathrm{Re}} + x_{j\,\mathrm{Im}} \quad y_{j\,\mathrm{Re}} + y_{j\,\mathrm{Im}} \quad z_{j\,\mathrm{Re}} + z_{j\,\mathrm{Im}} \rfloor,\ j = 0..n.$

Розглянемо частковий випадок, а саме підставимо у рівняння (2.6) $n = 3$, тобто у якості напрямної будемо застосовувати кубічну ізотропну криву Без'є. Одержимо:

$$\begin{aligned}\mathbf{r}_{\mathrm{Re}}(u,v) = {} & \mathbf{r}_{0\,\mathrm{Re}}(1-3u+3u^2-3v^2-u^3+3uv^2) - \mathbf{r}_{0\,\mathrm{Im}}(-3v+6uv-3u^2v+ \\ & +v^3) - (-3\mathbf{r}_{1\,\mathrm{Re}}(1-2u+u^2-v^2)+3\mathbf{r}_{1\,\mathrm{Im}}(-2v+2uv))u + (-3\mathbf{r}_{1\,\mathrm{Im}}(1-2u+ \\ & +u^2-v^2)-3\mathbf{r}_{1\,\mathrm{Re}}(-2v+2uv))v - (-3\mathbf{r}_{2\,\mathrm{Re}}(1-u)-3\mathbf{r}_{2\,\mathrm{Im}}v)(u^2-v^2)+ \\ & +2(-3\mathbf{r}_{2\,\mathrm{Im}}(1-u)+3\mathbf{r}_{2\,\mathrm{Re}}v)uv + \mathbf{r}_{3\,\mathrm{Re}}(u^3-3uv^2) - \mathbf{r}_{3\,\mathrm{Im}}(3u^2v-v^3),\end{aligned} \tag{2.7}$$

Розглянемо різні способи формоутворення напрямної ізотропної кривої Без'є та визначимо рівняння мінімальних поверхонь.

Після підстановки у вираз (2.6) одержимо:

$$\begin{aligned}\mathbf{r}(u+iv) = {} & \mathrm{Re}[\mathbf{r}_0(1-u-iv)^3 + 3\mathbf{r}_1(1-u-iv)^2(u+iv) + 3\mathbf{r}_2(1-u-iv)(u+iv)^2 + \\ & +\mathbf{r}_3(u+iv)^3],\end{aligned}$$

де $\mathbf{r}_0\lfloor (a_0-a_2)i \quad a_0+a_2 \quad -a_1 i\rfloor,$

$\mathbf{r}_1\lfloor (a_0-a_2-a_3)i \quad a_0+a_2+a_3 \quad -a_1 i\rfloor,$

$\mathbf{r}_2\lfloor (a_0-a_2-2a_3)i \quad a_0+a_2+2a_3 \quad (a_3-a_1)i\rfloor,$

$\mathbf{r}_3\lfloor (a_0-a_2-2a_3)i \quad a_0+a_2+4a_3 \quad (3a_3-a_1)i\rfloor.$

Виділимо дійсну частину з одержаного рівняння:

$$
\begin{aligned}
x(u,v) &= (-a_{0\mathrm{Im}} + a_{2\mathrm{Im}})(1-3u+3u^2-3v^2-u^3+3uv^2) - (a_{0\mathrm{Re}} - \\
&- a_{2\mathrm{Re}})(-3v+6uv-3u^2v+v^3) + (3(-a_{0\mathrm{Im}} + a_{2\mathrm{Im}} + a_{3\mathrm{Im}})(1-2u+u^2 - \\
&- v^2) - 6(a_{0\mathrm{Re}} - a_{2\mathrm{Re}} - a_{3\mathrm{Re}})(-v+uv))u - (3(a_{0\mathrm{Re}} - a_{2\mathrm{Re}} - a_{3\mathrm{Re}})(1- \\
&- 2u+u^2-v^2) + 6(-a_{0\mathrm{Im}} + a_{2\mathrm{Im}} + a_{3\mathrm{Im}})(-v+uv))v + 3((-a_{0\mathrm{Im}} + \\
&+ a_{2\mathrm{Im}} + 2a_{3\mathrm{Im}})(1-u) + (a_{0\mathrm{Re}} - a_{2\mathrm{Re}} - 2a_{3\mathrm{Re}})v)(u^2-v^2) - 6(a_{0\mathrm{Re}} - \\
&- a_{2\mathrm{Re}} - 2a_{3\mathrm{Re}})(1-u) - (-a_{0\mathrm{Im}} + a_{2\mathrm{Im}} + 2a_{3\mathrm{Im}})v)uv + (-a_{\mathrm{Im}} + a_{2\mathrm{Im}} + \\
&+ 2a_{3\mathrm{Im}})(u^3-3uv^2) - (a_{0\mathrm{Re}} - a_{2\mathrm{Re}} - 2a_{3\mathrm{Re}})(3u^2v-v^3),
\end{aligned}
\tag{2.8}
$$

$$
\begin{aligned}
y(u,v) &= (a_{0\mathrm{Re}} + a_{2\mathrm{Re}})(1-3u+3u^2-3v^2-u^3+3uv^2) - (a_{0\mathrm{Im}} + \\
&+ a_{2\mathrm{Im}})(-3v+6uv-3u^2v+v^3) + (3(a_{0\mathrm{Re}} + a_{2\mathrm{Re}} + a_{3\mathrm{Re}})(1-2u+u^2 - \\
&- v^2) - 6(a_{0\mathrm{Im}} + a_{2\mathrm{Im}} + a_{3\mathrm{Im}})(-v+uv))u - (3(a_{0\mathrm{Im}} + a_{2\mathrm{Im}} + a_{3\mathrm{Im}})(1- \\
&- 2u+u^2-v^2) + 6(a_{0\mathrm{Re}} + a_{2\mathrm{Re}} + a_{3\mathrm{Re}})(-v+uv))v + (3(a_{0\mathrm{Re}} + a_{2\mathrm{Re}} + \\
&+ 2a_{3\mathrm{Re}})(1-u) + 3(a_{0\mathrm{Im}} + a_{2\mathrm{Im}} + 2a_{3\mathrm{Im}})v)(u^2-v^2) - 6((a_{0\mathrm{Im}} + \\
&+ a_{2\mathrm{Im}} + 2a_{3\mathrm{Im}})(1-u) - (a_{0\mathrm{Re}} + a_{2\mathrm{Re}} + 2a_{3\mathrm{Re}})v)uv + (a_{0\mathrm{Re}} + a_{2\mathrm{Re}} + \\
&+ 4a_{3\mathrm{Re}})(u^3-3uv^2) - (a_{0\mathrm{Im}} + a_{2\mathrm{Im}} + 4a_{3im})(3u^2v-v^3),
\end{aligned}
$$

$$
\begin{aligned}
z(u,v) &= a_{1\mathrm{Im}}(1-3u+3u^2-3v^2-u^3+3uv^2) + a_{1\mathrm{Re}}(-3v+6uv-3u^2v + \\
&+ v^3) + (3a_{1\mathrm{Im}}(1-2u+u^2-v^2) + 6a_{1\mathrm{Re}}(-v+uv))u - (-3a_{1\mathrm{Re}}(1-2u + \\
&+ u^2-v^2) + 6a_{1\mathrm{Im}}(-v+uv))v + 3((-a_{3\mathrm{Im}} + a_{1\mathrm{Im}})(1-u) + (a_{3\mathrm{Re}} - \\
&- a_{1\mathrm{Re}})v)(u^2-v^2) - 6((a_{3\mathrm{Re}} - a_{1\mathrm{Re}})(1-u) - (-a_{3\mathrm{Im}} + a_{1\mathrm{Im}})v)uv + \\
&+ (-3a_{3\mathrm{Im}} + a_{1\mathrm{Im}})(u^3-3uv^2) - (3a_{3\mathrm{Re}} - a_{1\mathrm{Re}})(3u^2v-v^3).
\end{aligned}
$$

2.4. Моделювання поверхонь на основі ізотропних кривих та квазіконформної заміни параметру

Виконаємо квазіконформну заміну: $t=ku+iv$ або $t=u+ikv$.

Розглянемо частковий випадок, а саме підставимо у рівняння (2.6) $n=3$, тобто у якості напрямної будемо застосовувати кубічну ізотропну криву Без'є. При заміні $t=u+ikv$ одержимо:

$$\mathbf{r}_{\mathrm{Re}}(u,v)=\mathbf{r}_{0\,\mathrm{Re}}(1-3u+3u^2-3v^2k^2-u^3+3uv^2k^2)-\mathbf{r}_{0\,\mathrm{Im}}(-3vk+6uvk-$$
$$-3u^2vk+v^3k^3)-(-3\mathbf{r}_{1\,\mathrm{Re}}(1-2u+u^2-v^2k^2)+3\mathbf{r}_{1\,\mathrm{Im}}(-2vk+2uvk))u+$$
$$+(-3\mathbf{r}_{1\,\mathrm{Im}}(1-2u+u^2-v^2k^2)-3\mathbf{r}_{1\,\mathrm{Re}}(-2vk+2uvk))vk-(-3\mathbf{r}_{2\,\mathrm{Re}}(1-u)-$$
$$-3\mathbf{r}_{2\,\mathrm{Im}}vk)(u^2-v^2k^2)+2(-3\mathbf{r}_{2\,\mathrm{Im}}(1-u)+3\mathbf{r}_{2\,\mathrm{Re}}vk)uvk+\mathbf{r}_{3\,\mathrm{Re}}(u^3-$$
$$-3uv^2k^2)-\mathbf{r}_{3\,\mathrm{Im}}(3u^2vk-v^3k^3). \qquad (2.9)$$

При заміні $t=ku+iv$ будемо мати:

$$\mathbf{r}_{\mathrm{Re}}(u,v)=\mathbf{r}_{0\,\mathrm{Re}}(1-3uk+3u^2k^2-3v^2-u^3k^3+3uv^2k)-\mathbf{r}_{0\,\mathrm{Im}}(-3v+6uvk-$$
$$-3u^2vk^2+v^3)-(-3\mathbf{r}_{1\mathrm{Re}}(1-2uk+u^2k^2-v^2)+3\mathbf{r}_{1\mathrm{Im}}(-2v+2uvk))uk+$$
$$+(-3\mathbf{r}_{1\mathrm{Im}}(1-2uk+u^2k^2-v^2)-3\mathbf{r}_{1\mathrm{Re}}(-2v+2uvk))v-(-3\mathbf{r}_{2\,\mathrm{Re}}(1-uk)-$$
$$-3\mathbf{r}_{2\,\mathrm{Im}}v)(u^2k^2-v^2)+2(-3\mathbf{r}_{2\,\mathrm{Im}}(1-uk)+3\mathbf{r}_{2\,\mathrm{Re}}v)uvk+\mathbf{r}_{3\mathrm{Re}}(u^3k^3-$$
$$-3uv^2k)-\mathbf{r}_{3\mathrm{Im}}(3u^2vk^2-v^3). \qquad (2.10)$$

Розрахуємо коефіцієнти першої та другої квадратичної форм.

Для $t=u+ikv$ коефіцієнти першої квадратичної форми:

$$E=9a_{3\,\mathrm{Re}}^2[2u^2v^2k^2+v^4k^4+2v^2k^2+1+2u^2+u^4]+9a_{3\,\mathrm{Im}}^2[v^4k^4+$$
$$+2v^2k^2+2v^2k^2u^2+1+2u^2+u^4],$$
$$G=9a_{3\,\mathrm{Re}}^2[u^4k^2+2u^2v^2k^4+2u^2k^2+v^4k^6+2v^2k^4+k^2]+$$
$$+9a_{3\,\mathrm{Im}}^2[2u^2v^2k^4+v^4k^6+2v^2k^4+u^4k^2+2u^2k^2+k^2],$$
$$F=0. \qquad (2.11)$$

Аналіз виразів (2.11) показує що $E\neq G\Rightarrow\ k^2E=G$. Розрахуємо коефіцієнти другої квадратичної форми:

$$L=\frac{6a_{3\,\mathrm{Im}}k(a_{3\,\mathrm{Re}}^2[2u^2v^2k^2+v^4k^4+2v^2k^2+1+2u^2+u^4]}{\sqrt{(a_{3\,\mathrm{Im}}^2+a_{3\,\mathrm{Re}}^2)^2(v^2k^2+1+u^2)^4k^2}}+$$
$$+\frac{a_{3\,\mathrm{Im}}^2[v^4k^4+2v^2k^2+2v^2k^2u^2+1+2u^2+u^4])}{\sqrt{(a_{3\,\mathrm{Im}}^2+a_{3\,\mathrm{Re}}^2)^2(v^2k^2+1+u^2)^4k^2}}, \qquad (2.12)$$

$$M = \frac{6a_{3\,\mathrm{Re}}k^2(a_{3\,\mathrm{Re}}^2[2u^2v^2k^2 + v^4k^4 + 2v^2k^2 + 1 + 2u^2 + u^4]}{\sqrt{(a_{3\,\mathrm{Im}}^2 + a_{3\,\mathrm{Re}}^2)^2(v^2k^2 + 1 + u^2)^4k^2}} +$$
$$+ \frac{a_{3\,\mathrm{Im}}^2[v^4k^4 + 2v^2k^2 + 2v^2k^2u^2 + 1 + 2u^2 + u^4])}{\sqrt{(a_{3\,\mathrm{Im}}^2 + a_{3\,\mathrm{Re}}^2)^2(v^2k^2 + 1 + u^2)^4k^2}},$$
$$N = -\frac{6a_{3\,\mathrm{Im}}k^3(a_{3\,\mathrm{Re}}^2[2u^2v^2k^2 + v^4k^4 + 2v^2k^2 + 1 + 2u^2 + u^4]}{\sqrt{(a_{3\,\mathrm{Im}}^2 + a_{3\,\mathrm{Re}}^2)^2(v^2k^2 + 1 + u^2)^4k^2}} +$$
$$+ \frac{a_{3\,\mathrm{Im}}^2[v^4k^4 + 2v^2k^2 + 2v^2k^2u^2 + 1 + 2u^2 + u^4])}{\sqrt{(a_{3\,\mathrm{Im}}^2 + a_{3\,\mathrm{Re}}^2)^2(v^2k^2 + 1 + u^2)^4k^2}}.$$

Аналізуючи вирази (2.12) одержуємо наступні співвідношення: $k^2L = -N$, $M = \frac{a_{3\,\mathrm{Re}}Lk}{a_{3\,\mathrm{Im}}}$. Знайдемо середню кривину поверхні за виразом (2.5). Для цього проаналізуємо вирази чисельника:

$$LG = -\frac{N}{k^2}Ek^2 = -NE, \quad MF = M \cdot 0 = 0. \tag{2.13}$$

Підставляючи (2.12) у (2.5) будемо мати *H=0.*

Тепер віднесемо цю поверхню до ліній кривини. Для цього середній коефіцієнт другої квадратичної форми повинен дорівнювати 0: *M=0.* Для виконання цієї умови $a_{3\,\mathrm{Re}} = 0$. Для віднесення поверхні до асимптотичних ліній необхідно прирівняти нулю два інших коефіцієнта: *L=0, N=0*, а *M≠0.* В цьому випадку одержимо: $a_{3\,\mathrm{Im}} = 0$.

Якщо провести аналогічні дослідження з квазіконформною заміною *t=ku+iv,* то одержимо нульове значення середньої кривини. Сформулюємо твердження.

При квазіконформній заміні параметру *t=ku+iv* або *t=u+ikv* в рівнянні ізотропної кривої будемо мати мінімальну поверхню, тобто середня кривина такої поверхні буде дорівнювати 0.

2.5. Висновки до розділу

Було розглянуто загальні теоритичні відомості, що стосуються моделювання мінімальних поверхонь та пояснено відповідні терміни. Розглянуто основні формули, необхідні для створення відповідного додатку системи. Проаналізовано декілька часткових випадків моделювання мінімальних поверхонь на основі ізотропних кривих з конформною та квазіконформною заміною параметра. Майже всі системи розроблені не на мові Python та не мають української локалізації.

3. ОГЛЯД ЗАСОБІВ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ

Аналізуючи поставлену задачу та методи її вирішення, було вирішено розроблювати програмний комплекс на основі десктопних технологій. Головною перевагою десктоп-застосунку перед іншими варіантами є його універсальність і можливість використання без необхідності підключатися до інтернету.

3.1 Опис інструментів розробки

Програмний комплекс побудований з використанням такого набору технологій: мова програмування Python, комплекс бібліотек для графічного інтерфейсу (GUI) PyQT5, інтерактивну графічну бібліотеку plotly.py

Python — інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

plotly.py - це інтерактивна бібліотека з відкритим вихідним кодом і на основі веб-переглядачів для Python

Побудований поверх plotly.js, plotly.py є бібліотекою високого рівня, декларативна графіка. plotly.js поставляється з більш ніж з 30 типами діаграм,

включаючи наукові діаграми, 3D-графіки, статистичні діаграми, SVG-карти, фінансові діаграми та багато іншого.

Qt - це набір крос-платформних бібліотек C ++, які реалізують API високого рівня для доступу до багатьох аспектів сучасних настільних і мобільних систем. Вони включають послуги розміщення та позиціонування, мультимедіа, підключення NFC та Bluetooth, веб-браузер на основі Chromium, а також традиційну розробку інтерфейсу.

PyQt5 - це повний набір прив'язок Python для Qt v5. Вона реалізована як більш ніж 35 модулів розширення і дозволяє використовувати Python як альтернативну мову розробки додатків для C++ на всіх підтримуваних платформах, включаючи iOS і Android.

PyQt5 також може бути вбудований у додатки, що базуються на C ++, щоб дозволити користувачам цих програм налаштувати або підвищити функціональність цих програм.

3.2 Обґрунтування вибору програмної реалізації

При проектуванні системи було вивчено та проаналізовано предметну область та вимоги замовника. Після ретельного аналізу було вирішено розроблювати програмний продукт, який заснований на кросплатформених десктоп-технологіях.

Технології були обрані за принципом зручності у використанні, відкритості вихідних кодів, актуальності в наш час та можливістю виконання на будь-якій операційній системі. Платформа PyQt надає змогу створювати програми на мові Python на будь-якій операційній системі. Фреймворк PyQT5 надає величезні можливості для створення десктоп-застосунків будь-якої складності, забезпечуючи при цьому велику швидкодію та надійність.

Бібліотеку Plotly.py було обрано через те, що за допомогою неї можливо створювати складні графічні інтерфейси, які легко модифікувати, тестувати та розширювати в подальших циклах розробки програмного забезпечення. Мова

програмування Python була обрана через те, що вона є високорівневою, інтерпретованою мовою загального призначення з динамічною типізацією, надзвичайно поширеною в різних сферах розробки, від Embedded (MicroPython) до Data Science, де взагалі є абсолютним лідером. Також треба зазначити, що Python має одне з найбільших співтовариств в сфері графічного моделювання, велику кількість популярних бібліотек для графічного моделювання, таких, як matplotlib, plotly.

Дані технології в сукупності дають змогу збудувати якісний та надійний продукт, який захищений від патентних позовів з боку розробників, бо всі ці технології покриті ліцензіями, які виключають таку можливість і надають доступ до вихідних кодів даних проектів.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Система моделювання мінімальних поверхонь на основі ізотропних кривих з квазіконформною заміною параметра буде складатися з чотирьох модулів, кожен з яких буде розбиватися на різну кількість функціональних підблоків. Головним модулем буде кабінет користувача, який одночасно буде елементом компонування системи. Така структура, дозволить легко та інтуїтивно користуватися системою за рахунок того, що модулі розташовуються у порядку, який зазвичай використовують для вирішення схожих задач.

На рисунку 4.1 наведена схема структури системи, на якій розташовані всі програмні модулі.

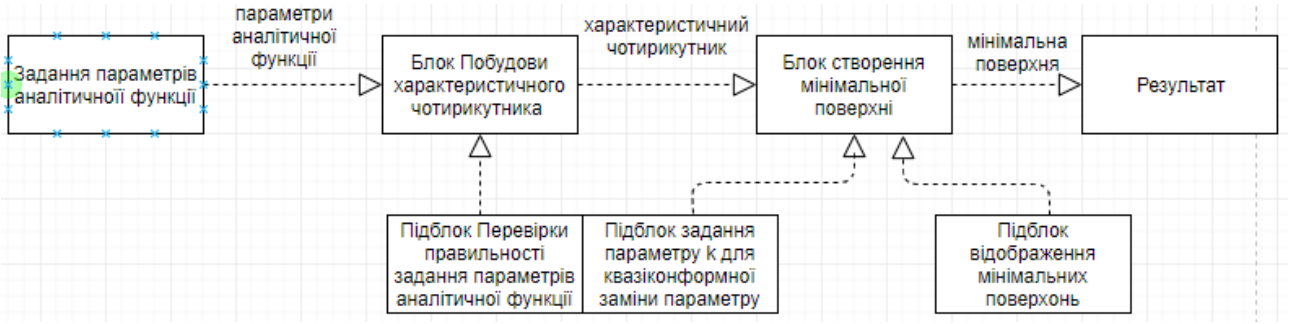

Рисунок 4.1 — Схема структури системи

4.1. Опис функціональності системи

Програмний застосунок для моделювання мінімальних поверхонь на основі ізотропних кривих містить у собі одного головного актора – користувач системи;

На рисунку 4.2 представлена діаграма прецедентів, яка описує функції та дії актора у системі.

Рисунок 4.2 — Діаграма прецедентів системи

4.2. Розробка кабінету користувача

Кабінет користувача – це основне робоче місце користувача в системі. Він включає в себе інші підмодулі, які виконують основні функції системи, та є елементом компонування в системі.

Підмодулі кабінету користувача:

До цих модулів належать наступні:

— модуль задання аналітичної функції;

— модуль розрахунку координат точок мінімальної поверхні;

— модуль відображення мінімальних поверхонь.

Кожен модуль вирішує відповідне завдання або проблему, при розробці власного додатку моделювання мінімальних поверхонь на основі ізотропних кривих з квазіконформною заміною параметра. В пунктах 4.3 – 4.5 цього розділу наведено детальний опис кожної з частин.

На рисунку 4.3 наведено вигляд проекту в результаті розробки.

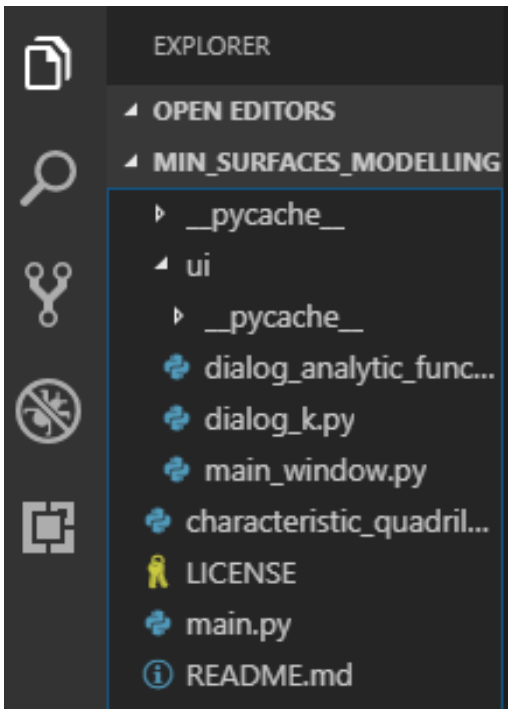

Рисунок 4.3 – Дерево проекту програмного додатку

Проект складається з головної форми main_window, діалогових вікон dialog_analytic_function, dialog_k для задання коефіцієнтів аналітичної функції та параметру k у випадку з квазіконформною заміною параметра, класу для обробки та перевірки введених значень з відповідних форм, класу для побудови характеристичного чотирикутника напрямної ізотропної кривої Безьє третього порядку, розрахованого за допомогою попередньо заданих коефіцієнтів аналітичної функції. Класи згруповані та взаємодіють між собою за шаблоном проектування MVC.

На головній формі розташовуються керуючі кнопки та відповідна інформація щодо задання необхідних параметрів та роботи з програмним додатком. Кнопкам для здійснення відображення та створення мінімальних поверхонь з конформною або квазіконформною заміною параметра попередньо встановлено властивість Disabled, тобто спочатку вони не активовані. Отже, ми маємо пубудувати відповідний характеристичний чотирикутник ізотропної кривої Безьє за відповідно до заданих коефіцієнтів аналітичної функції.

Далі ми маємо змогу створити мінімальну поверхню з конформною або квазіконформною заміною параметру, де, у випадку з квазіконформною

заміною маємо задати відповідне значення параметру k. Після здійснення побудови, активуються кнопки для відображення попередньо змодельованих мінімальних поверхонь, останньої або відразу декількох. Кнопка закриття додатку розташовується у правій верхній частині.

Для меню та області керування системою встановлено тему “Fusion“. Надписи чорного, зеленого та червоного кольорів, шрифт “Times New Roman“. Робоча область програми має традиційний дизайн, біла область та картинки для покращення зручності та інтуїтивно зрозумілого керування системою.

4.2.1. Шаблон проектування MVC

Model-View-Controller (зазвичай відомий як MVC) - це архітектурний паттерн, який зазвичай використовується для розробки інтерфейсів користувача, що розділяє програму на три взаємопов'язані частини. Це робиться для відокремлення внутрішніх уявлень інформації від способів подання інформації та прийняття від користувача. Шаблон дизайну MVC відокремлює ці основні компоненти, що дозволяє повторне використання коду та паралельний розвиток.

Традиційно використовувана для графічних інтерфейсів користувача (GUI), ця архітектура стала популярною для розробки веб-додатків. Популярні мови програмування, такі як JavaScript, Python, Ruby, PHP, Java і C # мають рамки MVC, які використовуються у розробці веб-додатків прямо з коробки.

MVC - це більше архітектурна схема, а не повне застосування. MVC в основному відноситься до UI / шару взаємодії програми. Вам все ще знадобиться рівень логіки бізнесу, можливо, рівень обслуговування та рівень доступу до даних.

Більшість додатків сьогодні слідують цій схемі, багато хто з невеликими варіаціями. Наприклад, деякі програми поєднують вигляд і контролер в один клас, оскільки вони вже дуже щільно зв'язані. Всі зміни сильно стимулюють

відокремлення даних та їх подання. Це не тільки спрощує структуру програми, але й дозволяє повторне використання коду.

Компоненти:

- **Модель**

Центральний компонент картини. Це динамічна структура даних програми, незалежна від інтерфейсу користувача. Він безпосередньо керує даними, логікою і правилами застосування.

- **Представлення**

Будь-яке представлення інформації, наприклад, діаграми, діаграми або таблиці. Можливі декілька переглядів однієї й тієї ж інформації, наприклад, діаграма для управління та табличний перегляд для бухгалтерів.

- **Контролер**

Приймає вхідні дані та перетворює їх у команди для моделі або перегляду.

Окрім розбиття програми на ці компоненти, конструкція моделі – перегляд-контролер визначає взаємодію між ними.

Модель відповідає за управління даними програми. Він отримує вхід користувача від контролера.

Вид означає представлення моделі в певному форматі.

Контролер реагує на вхід користувача і виконує взаємодії на об'єктах моделі даних. Контролер отримує вхід, додатково перевіряє його, а потім передає вхідні дані моделі.

Як і в інших моделях програмного забезпечення, MVC висловлює "ядро рішення" до проблеми, дозволяючи при цьому адаптуватися для кожної системи.

Цілі MVC:

- Одночасний розвиток

Оскільки MVC відокремлює різні компоненти програми, розробники можуть працювати паралельно на різних компонентах, не впливаючи або блокуючи один одного. Наприклад, команда може розділити своїх розробників між front-

end і back-end. Розробники back-end можуть конструювати структуру даних і як користувач взаємодіє з ним, не вимагаючи завершення інтерфейсу користувача. І навпаки, розробники інтерфейсу можуть розробляти та перевіряти макет програми до того, як буде доступна структура даних.

- Повторне використання коду

Одне і те ж (або подібне) представлення для однієї програми може бути використане для іншої програми з різними даними, оскільки перегляд просто обробляє, як дані відображаються користувачеві. На жаль, це не спрацьовує, коли цей код також використовується для обробки введеного користувачем. Наприклад, код DOM (включаючи спеціальні абстракції програми до нього) корисний як для графічного дисплея, так і для входу користувача. (Зауважте, що, незважаючи на назву Document Object Model, DOM насправді не є моделлю MVC, тому що це інтерфейс програми для користувача).

Щоб вирішити ці проблеми, MVC (і моделі, подібні до неї) часто поєднуються з архітектурою компонентів, яка забезпечує набір елементів інтерфейсу. Кожен елемент інтерфейсу є єдиним компонентом вищого рівня, який об'єднує 3 необхідні компоненти MVC в один пакет. Створюючи ці компоненти вищого рівня, які не залежать один від одного, розробники можуть повторно використовувати компоненти швидко і легко в інших додатках.

Переваги:

- Синхронна розробка - Кілька розробників можуть працювати одночасно на моделі, контролері і переглядах.
- Високе щеплення - MVC дозволяє логічне групування відповідних дій на контролері разом. Погляди для конкретної моделі також згруповані разом.
- Низька зв’язність - Сама природа рамки MVC така, що існує низька зв'язок між моделями, поглядами або контролерами
- Легкість модифікації - Через поділ обов'язків, майбутній розвиток або модифікація простіше

- Кілька переглядів для моделі - Моделі можуть мати декілька переглядів

Недоліки:

Недоліки MVC можна в цілому класифікувати як накладні витрати на некоректно розроблене програмне забезпечення.

- Навігація за кодами - навігація фреймворків може бути складною, оскільки вона запроваджує нові рівні абстракції і вимагає, щоб користувачі адаптувалися до критеріїв декомпозиції MVC.
- Послідовність мульти артефактів - розкладання функції на три артефакти викликає розсіювання. Таким чином, вимагають від розробників збереження узгодженості декількох подань одночасно.
- Підвержені неминучій кластеризації - додатки, як правило, мають важку взаємодію між тим, що бачить користувач і тим, що користувач використовує. Тому обчислення кожної функції і стан, як правило, скомпоновані в одну з 3 частин програми, видаляючи передбачувані переваги MVC.
- Надмірний шаблон - Через те, що обчислення та стан додатків зазвичай групуються в одну з трьох частин, інші частини вироджуються або в корінні шаблони, або за кодом, що існує лише для задоволення шаблону MVC.
- Висвітлена крива навчання - Знання про множинні технології стає нормою. Розробники, що використовують MVC, повинні бути кваліфікованими в декількох технологіях.
- Відсутність додаткових переваг - додатки користувальницького інтерфейсу вже враховані в компонентах і досягають повторного використання коду та незалежності через архітектуру компонентів, не залишаючи додаткових переваг для MVC.

На рисунку (4.4) наведено схему зв’язків між класами проекту, що використовує архітектурний шаблон MVC.

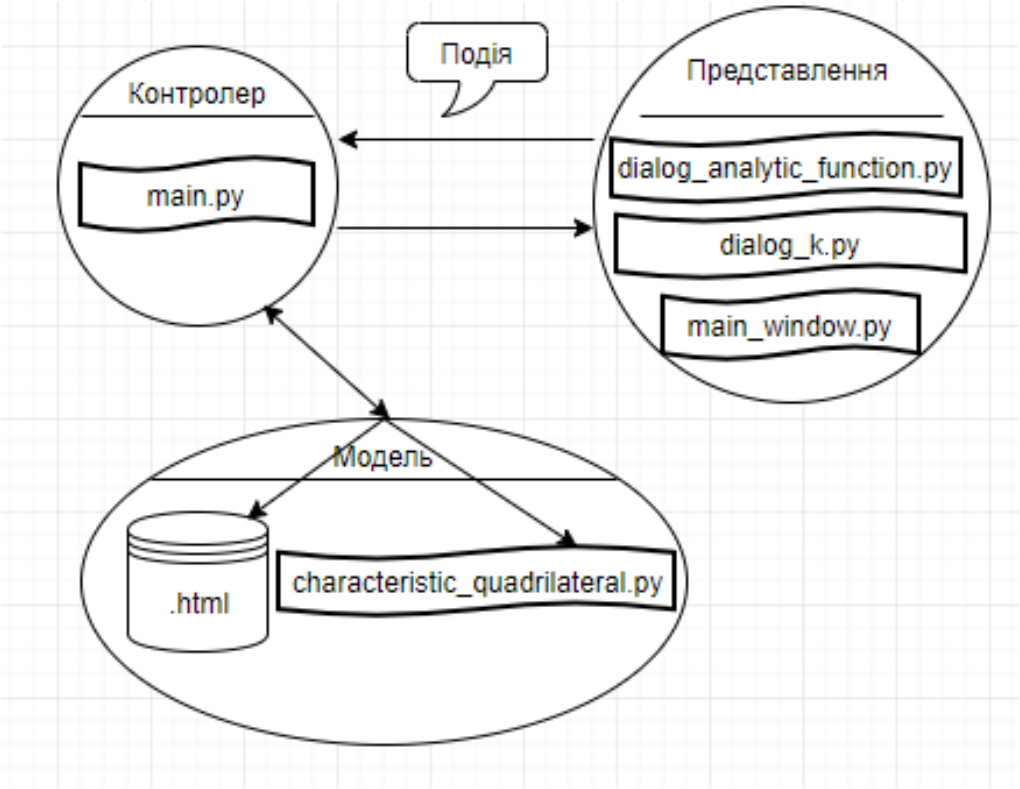

Рисунок 4.4 – Схема зв’язків між класами проекту

У файлах *.html зберігаються змодельовані мінімальні поверхні. Клас characteristic_quadrilateral.py зберігає логіку побудови характеристичного чотирикутнака напрямної ізотропної кривої Безьє та функції для побудови мінімальних поверхонь. Файли представлення зберігають логіку відображення графічного інтерфейсу системи. Файл контролеру реалізовує зв`язок файлів моделі та представлення. Зв’язок моделі та представлення здійснюється через контролер, тому шаблон проектування MVC реалізовує архітектурний патерн низька зв`язність, високе щеплення.

4.2.2. Низька зв`язність

Модулі повинні бути максимально незалежними від інших модулів, так що зміни в модулі не будуть сильно впливати на інші модулі.

Високі зв'язки означатимуть, що ваш модуль знає, як занадто багато інформації про внутрішні роботи інших модулів. Модулі, які знають занадто багато про інші модулі, роблять зміни важкими для координації і роблять

модулі крихкими. Якщо модуль А знає занадто багато про модуль В, зміни до модулів В можуть порушити функціональність модуля А.

Прагнучи до низької зв’язності, ви можете легко вносити зміни до внутрішніх модулів, не турбуючись про їх вплив на інші модулі системи. Низький зв'язок також полегшує розробку, записування та тестування кодів, оскільки наші модулі не взаємозалежні один від одного. Ми також отримуємо вигоду від легкого для повторного використання та складових модулів. Проблеми також відокремлені від невеликих автономних одиниць коду.

4.2.3 Високе щеплення

Щеплення часто відноситься до того, як елементи модуля розташовані разом. Пов'язаний код повинен бути близьким один до одного, щоб зробити його дуже щепленим.

Легко підтримуваний код зазвичай має високе щеплення. Елементи в модулі безпосередньо пов'язані з функціональними можливостями, які передбачено модулем. Підтримуючи високе щеплення у нашому коді, ми в кінцевому підсумку спробуємо зменшити дублювання знань в наших модулях. Ми можемо легко проектувати, писати і перевіряти наш код, оскільки код для модуля розташований разом і працює разом.

Низьке щеплення означатиме, що код, який складається з деякої функціональності, розповсюджується по всій кодовій базі. Не лише важко виявити, який код пов'язаний з вашим модулем, але й ще дуже складно переходити між різними модулями і відстежувати весь код у вашій голові.

4.3. Модуль задання аналітичної функції

Даний модуль дозволяє користувачу додатку задавати аналітичну функцію, а саме:

— задавати коефіцієнти аналітичної функції;

— перевіряти правильність введення коефіцієнтів;

— розраховувати координати характеристичного чотирикутника;

У файлі контролера описані методи задання коефіцієнтів аналітичної функції. Коефіцієнти зчитуються з відповідних полів форми (рисунок 4.5), призначених для їх задання та перевіряються щодо правильності вводу. Введені коефіцієнти мають бути числом або комплексним числом. Потім створюється об`єкт класу CharacteristicQuadrilateral, в конструктор ініціалізації якого передаються відповідні параметри аналітичної функції. Далі в конструкторі ініціалізації відбувається розрахунок координат характеристичного чотирикутника напрямної ізотропної кривої Безьє третього порядку. (рисунок 4.6)

Наступна функція виконує зчитування координат аналітичної функції з відповідних полів форми:

```
def catch_not_complex_number_error(self, value):
    try:
        number = complex(value)
        return number
    except Exception:
        QtWidgets.QMessageBox(self).about(self, 'Помилка','Введене значення може бути лише числом/комплексним числом')


def process_input(self):
    for i in range(len(self.all_line_edits)):
        if self.all_line_edits[i].isEnabled():
            self.characteristic_quadrilateral_coordinates[i] = self.catch_not_complex_number_error(self.all_line_edits[i].text())
            if self.characteristic_quadrilateral_coordinates[i]:
                self.all_line_edits[i].setEnabled(False)


    if all(elem is not None for elem in self.characteristic_quadrilateral_coordinates):
        self.parent.quadrilateral = CharacteristicQuadrilateral(self.characteristic_quadrilateral_coordinates)
        if not self.parent.ui.pushButton_3.isEnabled():
            self.parent.ui.pushButton_3.setEnabled(True)
        if not self.parent.ui.pushButton_4.isEnabled():
            self.parent.ui.pushButton_4.setEnabled(True)
        self.close()
```

Рисунок 4.5 – Введення та перевірка коефіцієнтів аналітичної функції

Спочатку відбувається перевірка введених до відповідних полів форми коефіцієнтів аналітичної функції, далі поля правильно введених значень блокуються. Якщо всі значення введені правильно, тобто є числами, або комплексними числами, відбувається створення об`єкта класу характеристичного чотирикутника.

Ініціалізація та розрахунок координат характеристичного чотирикутника за попередньо введеними коефіцієнтами аналітичної функції:

```
class CharacteristicQuadrilateral:
    def __init__(self, a):
        self.x0 = (a[0] - a[2])*1j
        self.y0 = a[0] + a[2]
        self.z0 = -a[1]*1j
        self.x1 = (a[0]-a[2]-a[3])*1j
        self.y1 = a[0]+a[2]+a[3]
        self.z1 = -a[1]*1j
        self.x2 = (a[0] - a[2] - 2*a[3])*1j
        self.y2 = a[0] + a[2] + 2*a[3]
        self.z2 = (a[3] - a[1])*1j
        self.x3 = (a[0] - a[2] - 2*a[3])*1j
        self.y3 = a[0] + a[2] + 4*a[3]
        self.z3 = (3*a[3] - a[1])*1j
```

Рисунок 4.6 – Ініціалізація полів класу CharacteristicQuadrilateral

Нижче, на рисунку 4.7 наведено блок-схему методів для зчитування та здійснення перевірки коефіцієнтів аналітичної функції, створення об’єкту класу характеристичного чотирикутника.

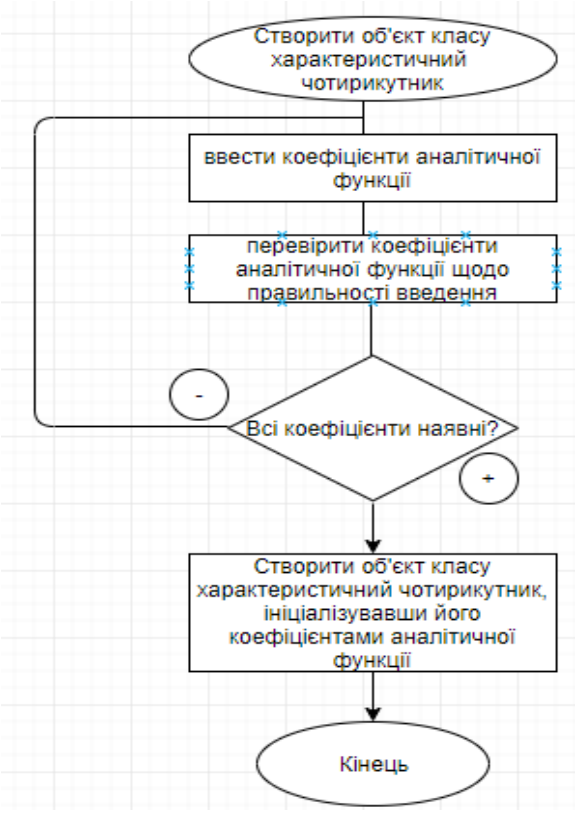

Рисунок 4.7 – Блок-схема методів зчитування та перевірки коефіцієнтів аналітичної функції

Після створення, зберігаємо об`єкт класу CharacteristicQuadrilateral в полі quadrilateral головного класу контролера.

4.4. Модуль побудови мінімальної поверхні

Даний модуль виконує всю роботу пов’язану з моделюванням мінімальної поверхні на основі ізотропних кривих, а саме:

— Розрахунок координат точок мінімальної поверхні на основі ізотропних кривих з конформною заміною параметра;

— Розрахунок координат точок мінімальної поверхні на основі ізотропних кривих з квазіконформною заміною параметра;

— Перевірку отриманої поверхні на «мінімальність» за допомогою попередньо розрахованої середньої кривизни поверхні на основі коефіцієнтів першої та другої квадратичних форм.

У класі контролера описані методи, що викликають відповідні методи моделювання мінімальних поверхонь на основі ізотропних кривих з конформною або квазіконформною заміною параметра для попередньо створеного об’єкту класу CharacteristicQuadrilateral.

На рисунку 4.8 наведено метод контролеру для створення діалогу побудови мінімальної поверхні у випадку з квазіконформною заміною параметра:

```
def open_create_minimal_surface_quasiconformal_replacement_dialog(self):
    dialog = DialogK(self)
    dialog.show()
```

Рисунок 4.8 – Створення діалогу введення параметру k, у випадку з квазіконформною заміною параметра

Після перевірки параметру k щодо правильності введення викликаємо відповідний метод об’єкту класу CharacteristicQuadrilateral для створення мінімальної поверхні у випадку з квазіконформною заміною параметра та додаємо щойно створену поверхню в масив всих створених за час роботи

додатку поверхонь. Активуємо кнопки для здійснення відображення щойно побудованих мінімальних поверхонь.

На рисунку 4.9 наведено основний метод для створення мінімальної поверхні з квазіконформною заміною параметра:

```
def process_input(self):
    self.parent.k = self.catch_not_number_error(self.ui.lineEdit.text())
    if self.parent.k:
        self.parent.current_fig = self.parent.quadrilateral.create_minimal_surface_quasiconformal_replacement("Мінімальна поверхня на основі
        self.parent.fig_list.append(self.parent.current_fig)
        if not self.parent.ui.pushButton_2.isEnabled(): self.parent.ui.pushButton_2.setEnabled(True)
        if (not self.parent.ui.pushButton.isEnabled()) and (len(self.parent.fig_list) > 1): self.parent.ui.pushButton.setEnabled(True)
        self.close()
```

Рисунок 4.8 – Головний метод контролеру для створення мінімальної поверхні у випадку з квазіконформною заміною параметра

На рисунках 4.8, 4.9 показано методи класу CharacteristicQuadrilateral, що викликатимуться методом контролера для створення мінімальної поверхні з квазіконформною заміною параметра.

```
def create_minimal_surface_quasiconformal_replacement(self, name, k):
    u,v = self.get_uv(0,1)
    x = self.quasiconformal_replacement(u, v, k, self.x0,self.x1,self.x2,self.x3)
    y = self.quasiconformal_replacement(u, v, k, self.y0,self.y1,self.y2,self.y3)
    z = self.quasiconformal_replacement(u, v, k, self.z0,self.z1,self.z2,self.z3)
    return self.create_trisurf_with_parameters(u, v, x, y, z, name)
```

Рисунок 4.8 – Метод класу CharacteristicQuadrilateral для розрахунку координат мінімальної поверхні, використовуючи координати характеристичного чотирикутника

```
def quasiconformal_replacement(self, u, v, k, r0, r1, r2, r3):
    return r0.real*(1 - 3*u + 3*u**2 - 3*v**2*k**2 - u**3 + 3*u*v**2*k**2) - r0.imag*(-3*v*k+6*u*v*k - 3*u**2*v*k + v**3*k**3) - \
                (-3*r1.real*(1 - 2*u + u**2 - v**2*k**2) + 3*r1.imag*(-2*v*k + 2*u*v*k))*u + (-3*r1.imag*(1 - 2*u+ u**2 - v**2*k**2) - \
                    3*r1.real*(-2*v*k+2*u*v*k))*v*k - (-3*r2.real*(1 - u) - 3*r2.imag*v*k)*(u**2 - v**2*k**2) + 2*(-3*r2.imag*(1 - u) + \
                        3*r2.real*v*k)*u*v*k + r3.real*(u**3 - 3*u*v**2*k**2) - r3.imag*(3*u**2*v*k - v**3*k**3)
```

Рисунок 4.9 – Метод, що задає формулу для випадку з квазіконформною заміною параметра

Рисунок 4.10 наводить лістинг основного методу контролеру для створення мінімальної поверхні у випадку з конформною заміною параметра:

```
def create_minimal_surface_conformal_replacement(self):
    self.current_fig = self.quadrilateral.create_minimal_surface_conformal_replacement("Мінімальна поверхня на основі аналітичної функції та
    self.fig_list.append(self.current_fig)
    if not self.ui.pushButton_2.isEnabled(): self.ui.pushButton_2.setEnabled(True)
    if (not self.ui.pushButton.isEnabled()) and (len(self.fig_list) > 1): self.ui.pushButton.setEnabled(True)
    self.ui.pushButton_4.setEnabled(False)
```

Рисунок 4.10 – Лістинг основного методу контролеру для створення мінімальної поверхні з конформною заміною параметра

На рисунках (4.11, 4.12) наведено відповідні методи класу CharacteristicQuadrilateral, що реалізують створення мінімальної поверхні у випадку з конформною заміною параметра:

```
def create_minimal_surface_conformal_replacement(self, name):
    u,v = self.get_uv(0,1)
    x = self.conformal_replacement(u, v, self.x0,self.x1,self.x2,self.x3)
    y = self.conformal_replacement(u, v, self.y0,self.y1,self.y2,self.y3)
    z = self.conformal_replacement(u, v, self.z0,self.z1,self.z2,self.z3)
    return self.create_trisurf_with_parameters(u, v, x, y, z, name)
```

Рисунок 4.11 – Лістинг методу класу характеристичного чотирикутника для розрахунку координат мінімальної поверхні.

```
def conformal_replacement(self, u, v, r0, r1, r2, r3):
    return (r0*(1-u-v*1j)**3+3*(r1*(1-u-v*1j)**2)*(u+v*1j)+3*(r2*(1-u-v*1j))*(v*1j+u)**2+r3*(u+v*1j)**3).real
```

Рисунок 4.12 – Метод, що задає формулу для випадку з конформною заміною параметра.

На рисунку 4.13 подано лістинг методу класу CharacteristicQuadrilateral для побудови мінімальної поверхні за попередньо розрахованими координатами:

```
def create_trisurf_with_parameters(self, u, v, x, y, z, name):
    points2D = np.vstack([u, v]).T
    tri = Delaunay(points2D)
    simplices = tri.simplices
    return FF.create_trisurf(x=x, y=y, z=z,
                             colormap=['rgb(50, 0, 75)', 'rgb(200, 0, 200)', '#c8dcc8'],
                             show_colorbar=True,
                             simplices=simplices,
                             title=name)
```

Рисунок 4.13 – Метод, що будує мінімальну поверхню, використовуючи задані параметри та координати мінімальної поверхні

Після здійснення побудов мінімальних поверхонь, перевіряємо отримані поверхні на мінімальність, рахуючи середню кривизну поверхні, використовуючи попередньо розраховані коефіцієнти першої та другої квадратичних форм. Нижче, на рисунку 4.14 наведені блок-схеми побудови

мінімальних поверхонь у випадку з конформною та квазіконформною заміною параметра.

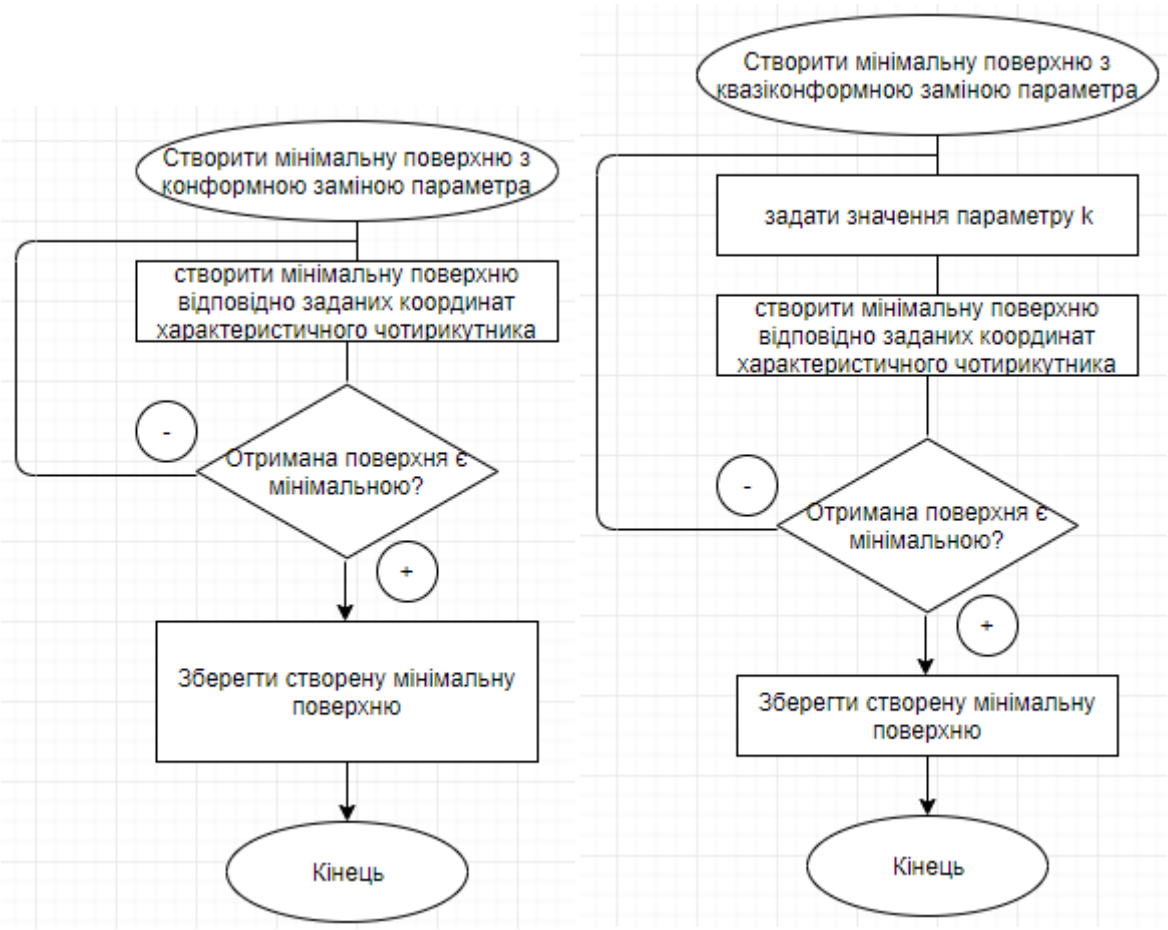

Рисунок 4.14 – Блок-схеми створення мінімальних поверхонь з конформною та квазіконформною заміною параметра відповідно.

4.5. Модуль відображення мінімальних поверхонь

Даний модуль будує та здійснює відображення попередньо створених в додатку мінімальних поверхонь, а саме:

— останню побудовану мінімальну поверхню;

— одразу всі створені в системі поверхні

Після створення мінімальних поверхонь даний модуль надає можливість здійснювати їх відображення, застосовуючи відповідний метод бібліотеки plotly.py. Описано методи для відображення останньої створеної в системі мінімальної поверхні та відразу всіх, створених в системі поверхонь.

На рисунках 4.15, 4.16 подані методи контролеру для відображення останньої, створеної в системі мінімальної поверхні та відразу всіх поверхонь відповідно:

```
def plot(self):
    py.plot(self.current_fig, "Мінімальна поверхня на основі аналітичної функції")
```

Рисунок 4.14 – Лістинг відображення останньої, створеної в системі мінімальної поверхні

```
def plot_all(self):
    data = []
    data_layout = []
    for fig in self.fig_list:
        data.append(fig.data[0])
        data_layout.append(fig.data[1])
    data += data_layout
    fig = go.Figure(data=data)
    py.plot(fig, "Мінімальна поверхня на основі аналітичної функції")
```

Рисунок 4.15 – Лістинг відображення одразу всіх, створених в системі мінімальних поверхонь

У випадку відображення відразу всіх поверхонь, циклічно акумулюємо дані всіх створених мінімальних поверхонь та дані їх відображення, після чого створюєму фігуру на основі попередньо акумулюваних даних та передаємо в функцію відображення бібліотеки plotly.py.

4.6. Локалізація інтерфейсу користувача

Одним із недоліків існуючих систем для моделювання мінімальних поверхонь є відсутність будь-якої локалізації інтерфейсу, крім англійської. Переклад інтерфейсу користувача українською мовою дозволить збільшити комфорт та ефективність роботи для користувача системи та знизити рівень знань, необхідних для роботи із програмним додатком.

Локалізація реалізовується за допомогою метода retranslateUi графічного фреймворку PyQt5.

На рисунках 4.16-4.18 наведено методи локалізації інтерфейсу користувача для кожної форми графічного інтерфейсу PyQt5:

```
def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow", "Головне Меню"))
    self.pushButton.setText(_translate("MainWindow", "Відобразити все"))
    self.pushButton_2.setText(_translate("MainWindow", "Відобразити"))
    self.pushButton_3.setText(_translate("MainWindow", "Створити"))
    self.pushButton_4.setText(_translate("MainWindow", "Створити"))
    self.pushButton2.setText(_translate("MainWindow", "Задати аналітичну функцію"))
    self.label.setText(_translate("MainWindow", "<html><head/><body><p><span style=\" color:#aa0000;\"> \
        Мінімальна поверхня з квазіконформною заміною параметра</span></p></body></html>"))
    self.label_2.setText(_translate("MainWindow", "<html><head/><body><p><span style=\" color:#005500;\"> \
        Мінімальна поверхня з конформною заміною параметра</span></p></body></html>"))
```

Рисунок 4.16 – Локалізація інтерфейсу головної форми

```
def retranslateUi(self, Dialog):
    _translate = QtCore.QCoreApplication.translate
    Dialog.setWindowTitle(_translate("Dialog", "Діалог"))
    self.label.setText(_translate("Dialog", "k = "))
```

Рисунок 4.17 – Локалізація інтерфейсу форми введення параметру k

```
def retranslateUi(self, Dialog):
    _translate = QtCore.QCoreApplication.translate
    Dialog.setWindowTitle(_translate("Dialog", "Аналітична функція"))
    self.label.setText(_translate("Dialog", "<html><head/><body><p><span style=\" font-size:10pt;\"> \
        Введіть коефіцієнти аналітичної функції: </span></p><p><br/></p></body></html>"))
```

Рисунок 4.18 – Локалізація інтерфейсу введення коефіцієнтів аналітичної функції

4.7 Висновки до розділу

Були розроблені модулі:

- задання аналітичної функції
- побудови мінімальної поверхні
- відображення мінімальних поверхонь

Досягнуто збільшення зручності користування завдяки наявності української локалізації, зручному та інтуїтивно зрозумілому інтерфейсу системи. Організовано точні розрахунки та перевірка створеної поверхні на мінімальність для усунення можливих помилок. Зменшено вимоги щодо ресурсів комп’ютера завдяки більш вузькому призначенні системи, на відміну від аналогів, тому розроблена система є набагато ефективнішою в використанні на відміну від існуючих програмних рішень.

5. МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Розроблена програмний комплекс розроблений з використанням кросплатформених десктоп-технологій і тому працює на основних операційних системах без необхідності мати інтернет доступ.

5.1. Інсталяція та системні вимоги

Застосунок працює з використанням десктоп-технологій, але так як це Portable версія, то не потребує встановлення, можливо запускати з будь-якого носія, наприклад, флешки.

5.2. Інструкція з використання програмного продукту

При вході на клієнтський додаток, нас вітає головне меню системи моделювання мінімальних поверхонь. На рисунку 5.1 відображене головне меню.

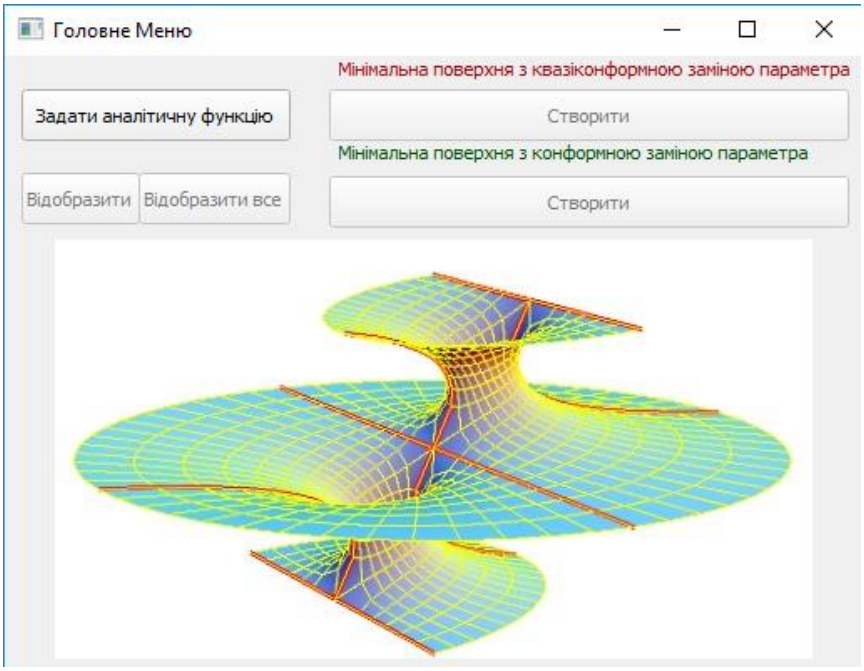

Рисунок 5.1 — Головне меню

Наступним кроком ми маємо задати аналітичну функцію, на основі коефіцієнтів якої буде розраховано координати характеристичного чотирикутника напрямної ізотропної кривої Безьє. На рисунку 5.2 зображена форма задання коефіцієнтів аналітичної функції.

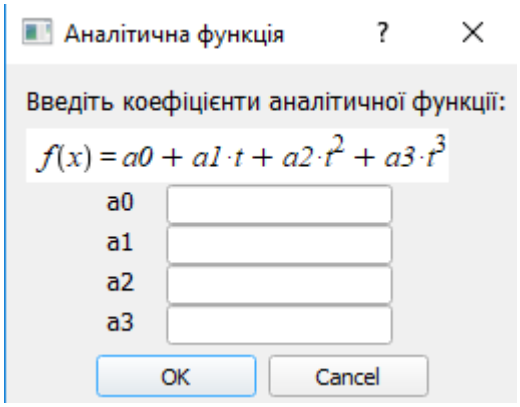

Рисунок 5.2 — Форма задання коефіцієнтів аналітичної ф-ї

Після перевірки введених значень коефіцієнтів аналітичної функції здійснюємо розрахунок координат характеристичного чотирикутника напрямної ізотропної кривої Безьє. Наступним кроком має бути здійснено розрахунок координат мінімальної поверхні на основі ізотропних кривих з конформною або квазіконформною заміною параметра. На рисунку 5.3 зображено діалог задання параметра k для випадку з квазіконформною заміною параметра.

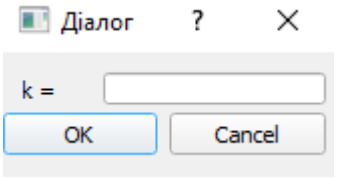

Рисунок 5.3 — Діалог задання параметру k

Далі в головному меню ми маємо змогу відобразити отриману мінімальну поверхню або створити інші поверхні. Кнопка «Відобразити» дозволяє відобразити останню змодельовану поверхню, кнопка «Відобразити всі» дозволяє відобразити всі наявні поверхні. Після відображення,

змодельовані поверхні зберігаються в форматі .html. На рисунках 5.4 та 5.5 відображено змодельовані мінімальні поверхні, 5.4 – мінімальна поверхня з конформною заміною параметра, задана аналітичною функцією, 5.5 – одразу дві мінімальні поверхні з квазіконформною заміною параметра.

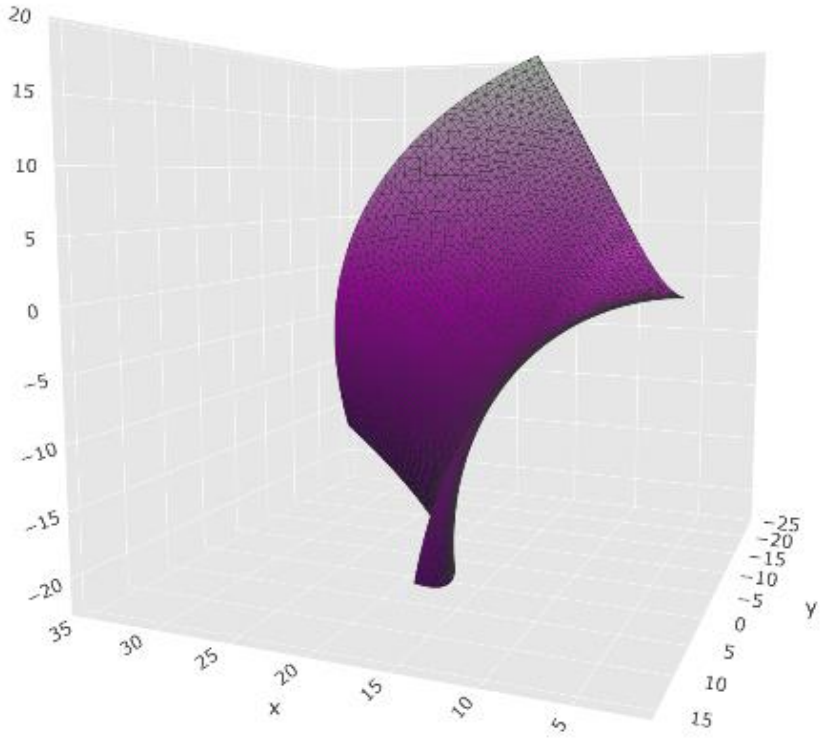

Рисунок 5.4 — Приклад відображення мінімальної поверхні на основі ізотропних кривих з конформною заміною параметра

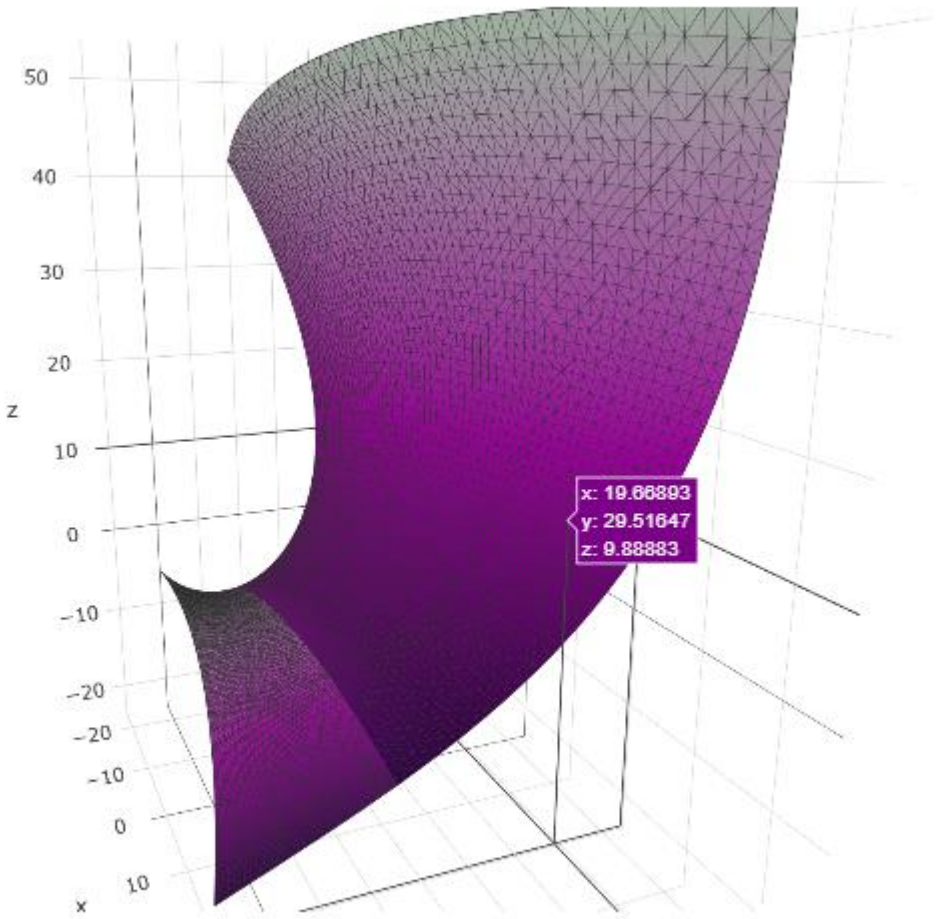

Рисунок 5.5 — Приклад відображення відразу двох мінімальних поверхонь з квазіконформною заміною параметру (k=-1.5, k =0.5)

ВИСНОВКИ

Результатом виконання дипломного проекту стала розробка програмного додатку для створення мінімальних поверхонь на основі ізотропних кривих з квазіконформною заміною параметра. Програмний продукт розроблений на мові програмування Python, з використанням графічного інтерфейсу користувача PyQt5 та інтерективною бібліотеки графічного відображення plotly.py.

У ході аналізу існуючого програмного забезпечення моделювання мінімальних поверхонь на основі ізотропних кривих з квазіконформною заміною параметра було досліджено системи, які слугують для вирішення поставлених задач. Аналіз показав, що існуючі системи вирішують задачу не у повному обсязі, є громіздкими та мають високі апаратні вимоги до пристроїв користувачів.

Розроблений програмний продукт дозволяє автоматизувати моделювання мінімальних поверхонь на основі ізотропних кривих з квазіконформною заміною параметра, надає зручний користувацький інтерфейс та можливість зберігати змодельовані поверхні в зручному форматі.

Проведено огляд методів і засобів розробки програмної системи. Обґрунтовано вибір створення програмної системи, заснованої на десктоп-технологіях, а також використано зручний архітектурний патерн MVC. Це дає змогу підвищити гнучкість та зручність системи, як у розробці та супроводі, так і у використані.

За результатами виконання тестових завдань підтверджена коректність отриманих результатів, отже система відповідає поставленим вимогам.

Користувачами системи можуть бути різноманітні користувачі, які здійснюють моделювання мінімальних поверхонь на основі ізотропних кривих з конформною та квазіконформною заміною параметра.

Програмне забезпечення може бути використано на будь-якій операційній системі, не потребує доступу до мережі інтернет, так як було розроблено з використанням крос-платформених десктоп-технологій.

Отже, практика покращила знання різноманітних технологій, що використовуються під час розробки програмного забезпечення. Також було досліджено різноманітні техніки та алгоритми графічного моделювання, було виявлено різноманітні сфери, для яких можна застосовувати дані прийоми, проведено аналіз існуючих бібліотек графічного моделювання та відображення. Також було створено декілька прототипів програмного забезпечення, які вирішували різноманітні аспекти поставленої задачі, а також які лягли в основу розробленого програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. CH Swaroop — A Byte of Python [Електронний ресурс]. — 2003. — Режим доступу: https://python.swaroopch.com/.

2. Markus Egger — MVVM Survival Guide for Enterprise Architectures in Silverlight and WPF [Електронний ресурс]. — 2012. — Режим доступу: https://www.packtpub.com/application-development/mvvm-survival-guide-enterprise-architectures-silverlight-and-wpf.

3. Martin Fowler — GUI Architectures. Часть 1 [Електронний ресурс]. — 2009. — Режим доступу: https://bit.ly/2CvCk1e.

4. Martin Fowler — GUI Architectures. Часть 2 [Електронний ресурс]. — 2009 — Режим доступу: https://habr.com/post/53536/.

5. П. Дейтел — Android для программистов. Создаем приложения / П. Дейтел — М.: Питер, 2012. — 560 с.

6. Голощапов Алексей — Google Android. Программирование для мобильных устройств / Алексей Голощапов. — М.: БХВ-Петербург, 2012. — 448 с.

7. Рик Роджерс — Android. Разработка приложений / Роджерс Рик. — М.: Эком, 2010. — 130 с.

8. Цехнер Марио — Программирование под Android / Марио Цехнер. — М.: Питер, 2012. — 688 с.

9. Левин Александр — Android на планшетах и смартфонах / Левин Александр. — М.: Питер, 2013. — 970 с.

10.Эккель Б. — Философия Java. 4-е полное издание [Електронний ресурс]. — 2015. — Режим доступу: http://www.twirpx.com/file/1606142/.

11.Хорстманн К. — Java SE8. Вводный курс / Кей Хорстманн. — М: Вильямс, 2014. — 208 с.

12.Блох Д. — Java. Эффективное программирование / Джошуа Блох — М. Лори, 2014. — 310 с.

13.Сиерра К., Бейтс Б. — Изучаем Java / Кэти Сиерра, Берт Бейтс —

М: Эксмо, 2016. — 720 с.

14.Больe А. — Learning SQL [Електронний ресурс]. — 2005. — Режим доступу: http://shop.oreilly.com/product/9780596007270.do.

15.Yuli Vasiliev — Beginning Database-Driven Application Development in Java EE. / Yuli Vasiliev. — Москва: ИЛ, 2008. — 400 с.

16.Генри С. Уоррен — Алгоритмические трюки для программистов / Генри С. Уоррен. — М.: Вильямс, 2014. — 512 с.

17.Гупта Арун — Java EE 7. Основы / Арун Гупта. - М.: Вильямс, 2014. — 336 с.

18.Эванс Б. — Java. Новое поколение разработки / Б. Эванс. — М.: Питер, 2014. — 400 с.

19. MichaelHerrmann. — PyQt Documentation. [Електронний ресурс]. — 2018. — Режим доступу: https://wiki.python.org/moin/PyQt.

ДОДАТОК А

Система моделювання мінімальних поверхонь на основі ізотропних кривих з квазіконформною заміною параметра

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТІ51193_ 18Б 12-2

Аркушів 1

Київ — 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ51193_18Б 79-1	Записка Кондратьєв.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТІ51193_18Б 12-1	server.pro	Серверний додаток
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТІ51193_ 18Б 12-2	client.pro	Клієнтський додаток

ДОДАТОК Б

Система моделювання мінімальних поверхонь на основі ізотропних кривих з квазіконформною заміною параметра

Текст програмного модуля

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТІ51193_ 18Б 12-2

Аркушів 5

Київ — 2019

// characteristic_quadrilateral.py

```
import plotly.figure_factory as FF
import numpy as np
from scipy.spatial import Delaunay


class CharacteristicQuadrilateral:
    def __init__(self, a):
        self.x0 = (a[0] - a[2])*1j
        self.y0 = a[0] + a[2]
        self.z0 = -a[1]*1j
        self.x1 = (a[0]-a[2]-a[3])*1j
        self.y1 = a[0]+a[2]+a[3]
        self.z1 = -a[1]*1j
        self.x2 = (a[0] - a[2] - 2*a[3])*1j
        self.y2 = a[0] + a[2] + 2*a[3]
        self.z2 = (a[3] - a[1])*1j
        self.x3 = (a[0] - a[2] - 2*a[3])*1j
        self.y3 = a[0] + a[2] + 4*a[3]
        self.z3 = (3*a[3] - a[1])*1j


    def create_trisurf_with_parameters(self, u, v, x, y, z, name):
        points2D = np.vstack([u, v]).T
        tri = Delaunay(points2D)
        simplices = tri.simplices
        return FF.create_trisurf(x=x, y=y, z=z,
                                 colormap=['rgb(50, 0, 75)', 'rgb(200, 0, 200)', '#c8dcc8'],
                                 show_colorbar=True,
                                 simplices=simplices,
                                 title=name)


    def get_uv(self, min, max):
        u=np.linspace(min, max, 60)
        v=np.linspace(min, max, 60)
        u,v=np.meshgrid(u,v)
        u=u.flatten()
        v=v.flatten()
        return u, v


    def conformal_replacement(self, u, v, r0, r1, r2, r3):
        return (r0*(1-u-v*1j)**3+3*(r1*(1-u-v*1j)**2)*(u+v*1j)+3*(r2*(1-u-v*1j))*(v*1j+u)**2+r3*(u+v*1j)**3).real


    def quasiconformal_replacement(self, u, v, k, r0, r1, r2, r3):
        return r0.real*(1 - 3*u + 3*u**2 - 3*v**2*k**2 - u**3 + 3*u*v**2*k**2) - r0.imag*(-3*v*k+6*u*v*k - 3*u**2*v*k + v**3*k**3) - \
                (-3*r1.real*(1 - 2*u + u**2 - v**2*k**2) + 3*r1.imag*(-2*v*k + 2*u*v*k))*u + (-3*r1.imag*(1 - 2*u+ u**2 - v**2*k**2) - \
                    3*r1.real*(-2*v*k+2*u*v*k))*v*k - (-3*r2.real*(1 - u) - 3*r2.imag*v*k)*(u**2 - v**2*k**2) + 2*(-3*r2.imag*(1 - u) + \
                        3*r2.real*v*k)*u*v*k + r3.real*(u**3 - 3*u*v**2*k**2) - r3.imag*(3*u**2*v*k - v**3*k**3)


    def create_minimal_surface_conformal_replacement(self, name):
        u,v = self.get_uv(0,1)
        x = self.conformal_replacement(u, v, self.x0,self.x1,self.x2,self.x3)
        y = self.conformal_replacement(u, v, self.y0,self.y1,self.y2,self.y3)
        z = self.conformal_replacement(u, v, self.z0,self.z1,self.z2,self.z3)
        return self.create_trisurf_with_parameters(u, v, x, y, z, name)


    def create_minimal_surface_quasiconformal_replacement(self, name, k):
        u,v = self.get_uv(0,1)
        x = self.quasiconformal_replacement(u, v, k, self.x0,self.x1,self.x2,self.x3)
        y = self.quasiconformal_replacement(u, v, k, self.y0,self.y1,self.y2,self.y3)
        z = self.quasiconformal_replacement(u, v, k, self.z0,self.z1,self.z2,self.z3)
        return self.create_trisurf_with_parameters(u, v, x, y, z, name)
```

// main.py

```
import sys
import plotly.offline as py
import plotly.graph_objs as go
```

```
from characteristic_quadrilateral import CharacteristicQuadrilateral
from ui.dialog_analytic_function import *
from ui.dialog_k import *
from ui.main_window import *


class DialogAnalyticFunction(QtWidgets.QDialog):
    def __init__(self, parent):
        QtWidgets.QWidget.__init__(self, parent)
        self.parent = parent
        self.ui = Ui_DialogAnalyticFunction()
        self.ui.setupUi(self)
        self.all_line_edits = [self.ui.lineEdit, self.ui.lineEdit_2, self.ui.lineEdit_3, self.ui.lineEdit_4]
        self.characteristic_quadrilateral_coordinates = [None] * 4
        self.ui.buttonBox.accepted.connect(self.process_input)


    def catch_not_complex_number_error(self, value):
        try:
            number = complex(value)
            return number
        except Exception:
            QtWidgets.QMessageBox(self).about(self, 'Помилка','Введене значення може бути лише числом/комплексним числом')


    def process_input(self):
        for i in range(len(self.all_line_edits)):
            if self.all_line_edits[i].isEnabled():
                self.characteristic_quadrilateral_coordinates[i] = self.catch_not_complex_number_error(self.all_line_edits[i].text())
                if self.characteristic_quadrilateral_coordinates[i]:
                    self.all_line_edits[i].setEnabled(False)


        if all(elem is not None for elem in self.characteristic_quadrilateral_coordinates):
            self.parent.quadrilateral = CharacteristicQuadrilateral(self.characteristic_quadrilateral_coordinates)
            if not self.parent.ui.pushButton_3.isEnabled():
                self.parent.ui.pushButton_3.setEnabled(True)
            if not self.parent.ui.pushButton_4.isEnabled():
                self.parent.ui.pushButton_4.setEnabled(True)
            self.close()


class DialogK(QtWidgets.QDialog):
    def __init__(self, parent):
        QtWidgets.QWidget.__init__(self, parent)
        self.parent = parent
        self.ui = Ui_DialogK()
        self.ui.setupUi(self)
        self.ui.buttonBox.accepted.connect(self.process_input)


    def catch_not_number_error(self, value):
        try:
            number = float(value)
            return number
        except Exception:
            QtWidgets.QMessageBox(self).about(self, 'Помилка','Введене значення може бути лише числом')


    def process_input(self):
        self.parent.k = self.catch_not_number_error(self.ui.lineEdit.text())
        if self.parent.k:
            self.parent.current_fig = self.parent.quadrilateral.create_minimal_surface_quasiconformal_replacement("Мінімальна поверхня на
основі аналітичної функції та квазікомфорної заміни параметру", self.parent.k)
            self.parent.fig_list.append(self.parent.current_fig)
            if not self.parent.ui.pushButton_2.isEnabled(): self.parent.ui.pushButton_2.setEnabled(True)
            if (not self.parent.ui.pushButton.isEnabled()) and (len(self.parent.fig_list) > 1): self.parent.ui.pushButton.setEnabled(True)
            self.close()


class MainWin(QtWidgets.QMainWindow):
    def __init__(self, parent=None):
        QtWidgets.QWidget.__init__(self, parent)
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.current_fig = None
```

```
        self.fig_list = []
        self.ui.pushButton2.clicked.connect(self.open_characteristic_quadrilateral_dialog)
        self.ui.pushButton_3.clicked.connect(self.open_create_minimal_surface_quasiconformal_replacement_dialog)
        self.ui.pushButton_4.clicked.connect(self.create_minimal_surface_conformal_replacement)
        self.ui.pushButton_2.clicked.connect(self.plot)
        self.ui.pushButton.clicked.connect(self.plot_all)


    def open_characteristic_quadrilateral_dialog(self):
        dialog = DialogAnalyticFunction(self)
        dialog.show()


    def open_create_minimal_surface_quasiconformal_replacement_dialog(self):
        dialog = DialogK(self)
        dialog.show()


    def create_minimal_surface_conformal_replacement(self):
        self.current_fig = self.quadrilateral.create_minimal_surface_conformal_replacement("Мінімальна поверхня на основі аналітичної
функції та комфорної заміни параметру")
        self.fig_list.append(self.current_fig)
        if not self.ui.pushButton_2.isEnabled(): self.ui.pushButton_2.setEnabled(True)
        if (not self.ui.pushButton.isEnabled()) and (len(self.fig_list) > 1): self.ui.pushButton.setEnabled(True)
        self.ui.pushButton_4.setEnabled(False)


    def plot(self):
        py.plot(self.current_fig, "Мінімальна поверхня на основі аналітичної функції")


    def plot_all(self):
        data = []
        data_layout = []
        for fig in self.fig_list:
            data.append(fig.data[0])
            data_layout.append(fig.data[1])
        data += data_layout
        fig = go.Figure(data=data)
        py.plot(fig, "Мінімальна поверхня на основі аналітичної функції")


if __name__=="__main__":
    app = QtWidgets.QApplication(sys.argv)
    app.setStyle('Fusion')
    myapp = MainWin()
    myapp.show()
    sys.exit(app.exec_())
```

//main_window.py

```
from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(513, 403)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.pushButton = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton.setEnabled(False)
        self.pushButton.setGeometry(QtCore.QRect(80, 70, 91, 31))
        self.pushButton.setObjectName("pushButton")
        self.pushButton_2 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_2.setEnabled(False)
        self.pushButton_2.setGeometry(QtCore.QRect(10, 70, 71, 31))
        self.pushButton_2.setObjectName("pushButton_2")
        self.pushButton_3 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_3.setEnabled(False)
        self.pushButton_3.setGeometry(QtCore.QRect(194, 20, 311, 31))
        self.pushButton_3.setObjectName("pushButton_3")
        self.pushButton_4 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_4.setEnabled(False)
        self.pushButton_4.setGeometry(QtCore.QRect(194, 72, 311, 31))
```

```
        self.pushButton_4.setObjectName("pushButton_4")
        self.pushButton2 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton2.setGeometry(QtCore.QRect(10, 20, 161, 31))
        self.pushButton2.setObjectName("pushButton2")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(200, 0, 311, 16))
        self.label.setObjectName("label")
        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(200, 50, 281, 16))
        self.label_2.setObjectName("label_2")
        self.label_3 = QtWidgets.QLabel(self.centralwidget)
        self.label_3.setGeometry(QtCore.QRect(30, 110, 461, 251))
        self.label_3.setText("")
        self.label_3.setPixmap(QtGui.QPixmap("C:/Users/arsen/Desktop/riemann.jpg"))
        self.label_3.setObjectName("label_3")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 513, 21))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)


    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Головне Меню"))
        self.pushButton.setText(_translate("MainWindow", "Відобразити все"))
        self.pushButton_2.setText(_translate("MainWindow", "Відобразити"))
        self.pushButton_3.setText(_translate("MainWindow", "Створити"))
        self.pushButton_4.setText(_translate("MainWindow", "Створити"))
        self.pushButton2.setText(_translate("MainWindow", "Задати аналітичну функцію"))
        self.label.setText(_translate("MainWindow", "<html><head/><body><p><span style=\" color:#aa0000;\"> \
            Мінімальна поверхня з квазіконформною заміною параметра</span></p></body></html>"))
        self.label_2.setText(_translate("MainWindow", "<html><head/><body><p><span style=\" color:#005500;\"> \
             Мінімальна поверхня з конформною заміною параметра</span></p></body></html>"))
```

//dialog_k.py

```
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_DialogK(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.resize(165, 62)
        self.buttonBox = QtWidgets.QDialogButtonBox(Dialog)
        self.buttonBox.setGeometry(QtCore.QRect(0, 30, 161, 21))
        self.buttonBox.setOrientation(QtCore.Qt.Horizontal)
        self.buttonBox.setStandardButtons(QtWidgets.QDialogButtonBox.Cancel|QtWidgets.QDialogButtonBox.Ok)
        self.buttonBox.setObjectName("buttonBox")
        self.label = QtWidgets.QLabel(Dialog)
        self.label.setGeometry(QtCore.QRect(10, 10, 51, 16))
        self.label.setObjectName("label")
        self.lineEdit = QtWidgets.QLineEdit(Dialog)
        self.lineEdit.setGeometry(QtCore.QRect(50, 10, 111, 16))
        self.lineEdit.setObjectName("lineEdit")
        self.retranslateUi(Dialog)
        self.buttonBox.rejected.connect(Dialog.reject)
        QtCore.QMetaObject.connectSlotsByName(Dialog)


    def retranslateUi(self, Dialog):
        _translate = QtCore.QCoreApplication.translate
        Dialog.setWindowTitle(_translate("Dialog", "Діалог"))
        self.label.setText(_translate("Dialog", "k = "))
```

//dialog_analytic_function

```
from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_DialogAnalyticFunction(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.resize(256, 169)
        self.buttonBox = QtWidgets.QDialogButtonBox(Dialog)
        self.buttonBox.setGeometry(QtCore.QRect(-130, 140, 341, 32))
        self.buttonBox.setOrientation(QtCore.Qt.Horizontal)
        self.buttonBox.setStandardButtons(QtWidgets.QDialogButtonBox.Cancel|QtWidgets.QDialogButtonBox.Ok)
        self.buttonBox.setObjectName("buttonBox")
        self.label = QtWidgets.QLabel(Dialog)
        self.label.setGeometry(QtCore.QRect(10, 10, 271, 16))
        self.label.setObjectName("label")
        self.lineEdit = QtWidgets.QLineEdit(Dialog)
        self.lineEdit.setGeometry(QtCore.QRect(80, 60, 113, 20))
        self.lineEdit.setText("")
        self.lineEdit.setObjectName("lineEdit")
        self.lineEdit_2 = QtWidgets.QLineEdit(Dialog)
        self.lineEdit_2.setGeometry(QtCore.QRect(80, 80, 113, 20))
        self.lineEdit_2.setObjectName("lineEdit_2")
        self.lineEdit_3 = QtWidgets.QLineEdit(Dialog)
        self.lineEdit_3.setGeometry(QtCore.QRect(80, 100, 113, 20))
        self.lineEdit_3.setObjectName("lineEdit_3")
        self.lineEdit_4 = QtWidgets.QLineEdit(Dialog)
        self.lineEdit_4.setGeometry(QtCore.QRect(80, 120, 113, 20))
        self.lineEdit_4.setObjectName("lineEdit_4")
        self.label_2 = QtWidgets.QLabel(Dialog)
        self.label_2.setGeometry(QtCore.QRect(50, 60, 47, 13))
        self.label_2.setObjectName("label_2")
        self.label_3 = QtWidgets.QLabel(Dialog)
        self.label_3.setGeometry(QtCore.QRect(50, 80, 47, 13))
        self.label_3.setObjectName("label_3")
        self.label_4 = QtWidgets.QLabel(Dialog)
        self.label_4.setGeometry(QtCore.QRect(50, 100, 47, 13))
        self.label_4.setObjectName("label_4")
        self.label_5 = QtWidgets.QLabel(Dialog)
        self.label_5.setGeometry(QtCore.QRect(50, 120, 47, 13))
        self.label_5.setObjectName("label_5")
        self.label_6 = QtWidgets.QLabel(Dialog)
        self.label_6.setGeometry(QtCore.QRect(10, 30, 211, 31))
        self.label_6.setText("")
        self.label_6.setPixmap(QtGui.QPixmap("C:/Users/arsen/Desktop/Screenshot_3.png"))
        self.label_6.setObjectName("label_6")
        self.retranslateUi(Dialog)
        self.buttonBox.rejected.connect(Dialog.reject)
        QtCore.QMetaObject.connectSlotsByName(Dialog)


    def retranslateUi(self, Dialog):
        _translate = QtCore.QCoreApplication.translate
        Dialog.setWindowTitle(_translate("Dialog", "Аналітична функція"))
        self.label.setText(_translate("Dialog", "<html><head/><body><p><span style=\" font-size:10pt;\"> \
            Введіть коефіцієнти аналітичної функції: </span></p><p><br/></p></body></html>"))
        self.label_2.setText(_translate("Dialog", "<html><head/><body><p><span style=\" font-size:10pt;\">a0</span></p></body></html>"))
        self.label_3.setText(_translate("Dialog", "<html><head/><body><p><span style=\" font-size:10pt;\">a1</span></p></body></html>"))
        self.label_4.setText(_translate("Dialog", "<html><head/><body><p><span style=\" font-size:10pt;\">a2</span></p></body></html>"))
        self.label_5.setText(_translate("Dialog", "<html><head/><body><p><span style=\" font-size:10pt;\">a3</span></p></body></html>"))
```

ДОДАТОК В

Система моделювання мінімальних поверхонь на основі ізотропних кривих з квазіконформною заміною параметра

Опис програмного модуля

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТІ51193_ 18Б 12-2

Аркушів 7

Київ — 2019

АНОТАЦІЯ

У додатку надана інформація про програмну частину системи моделювання мінімальних поверхонь на основі ізотропних кривих з квазіконформною заміною параметру.

Програмний продукт являє собою клієнтський додаток для ПК на базі Windows XP/7/8/10. Для створення було використано фреймворк PyQt5 та бібліотеку для інтерактивного графічного моделювання plotly.py.

Клієнтський додаток отримує коефіцієнти аналітичної функції для побудови характеристичного чотирикутника напрямної ізотропної кривої Безьє третього порядку та значення параметру k у випадку з квазіконформною заміною параметра. Далі відбувається моделювання відповідних мінімальних поверхонь щодо заданих параметрів з конформною або квазіконформною заміною.

ЗМІСТ

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Основним призначенням даного програмного забезпечення є надання користувачу можливості моделювання мінімальних поверхонь на основі ізотропних кривих та зручного збереження отриманих змодельованих поверхонь у зручному форматі .html.

Розроблений програмні модулі продукту мають такі функції:

1. Реалізовувати основний функціонал для здійснення моделювання мінімальних поверхонь
2. Робити точні розрахунки та графічне моделювання відповідно до заданих параметрів.
3. Мати вичерпний та інтуїтивно зрозумілий користувацький інтерфейс
4. Зберігати змодельовані мінімальні поверхні в зручному форматі

ОПИС СТРУКТУРИ СИСТЕМИ

Система моделювання мінімальних поверхонь на основі ізотропних кривих з квазіконформною заміною параметра буде складатися з чотирьох модулів, кожен з яких буде розбиватися на різну кількість функціональних підблоків. Головним модулем буде кабінет користувача, який одночасно буде елементом компонування системи. Така структура, дозволить легко та інтуїтивно користуватися системою за рахунок того, що модулі розташовуються у порядку, який зазвичай використовують для вирішення схожих задач.

Проект складається з головної форми main_window, діалогових вікон dialog_analytic_function, dialog_k для задання коефіцієнтів аналітичної функції та параметру k у випадку з квазіконформною заміною параметра, класу для обробки та перевірки введених значень з відповідних форм, класу для побудови характеристичного чотирикутника напрямної ізотропної кривої Безьє третього порядку, розрахованого за допомогою попередньо заданих коефіцієнтів аналітичної функції. Класи згруповані та взаємодіють між собою за шаблоном проектування MVC.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для створення системи було використано:

- Python;
- PyQt5;
- Plotly.py;

Розроблена програмний комплекс розроблений з використанням кросплатформених десктоп-технологій і тому працює на основних операційних системах без необхідності мати інтернет доступ.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Застосунок працює з використанням десктоп-технологій, але так як це Portable версія, то не потребує встановлення, можливо запускати з будь-якого носія, наприклад, флешки.

ВХІДНІ ТА ВИХІДНІ ДАНІ

Вхідні дані: коефіцієнти аналітичної функції, за домогою яких буде здійснено розрахунок координат характеристичного чотирикутнига ізотропної кривої Безьє третього порядку.

Вихідні дані: Змодельована мінімальна поверхня (поверхні) у форматі .html.